CD-ROM付

Windows95/98/NT4対応

10日でおぼえる Visual Basic 6.0 実践教室

VBテックラボ&瀬戸 遥 著

SE
SHOEISHA

本書の内容

本書は、『10日でおぼえるVisual Basic 6.0 入門教室』の続編です。Visual Basic 6.0を使って、はじめてWindowsのプログラムを作成しようとする人や、他のプログラミング言語を使ったことはあるが、Visual Basicを使うのははじめて、という方を対象に、すぐにVisual BasicでWindowsのプログラムを作成できるように、随所に工夫を凝らした構成で編集しました。

全体の流れは、「プログラミングセミナー」のように、段階を追ってVisual Basicの使い方やテクニックを説明してあります。各レッスンは4段階になっています。

【レッスンの構成】

1. サンプルの説明
サンプルの動作と、
作成の際のポイントを解説します。

2. 操作説明
作成に伴う一連の操作を、
わかりやすいように画面を使って
説明します。

3. コードの解説
プログラムコードを、
1行ずつ丁寧に解説していきます。

4. 練習問題
練習問題で基礎を固め、応用力をつけます。

紹介しているサンプルのコードは、入門者が混乱しないように、説明のポイントを重点的に絞って、なるべく少ない行数で作成しました。もっとも長いものでもコードの入力に時間を取られることはありません。また、これらのサンプルはコンパクトで理解しやすく、様々なテクニックを簡潔に紹介しているので、自分で開発する際に応用することができます。サンプルの完成バージョンは付属CD-ROMに収録されています。

本書では、『入門教室』を読了した読者が、より実用的なプログラムを作るためのヒントを幅広く紹介しています。従来のプログラミング解説書のように、プログラミングのテクニックに関する抽象的な概要や概念の説明ばかりで、プログラムを作る前からうんざりしてしまうようなことはありません。そういった「解説」はレッスンの最後にまとめて、まず「作ること」それ自体を楽しんでもらうことを主としたレッスン構成にしています。

本書は「作りながらおぼえたい人」と「おぼえてから作りたい人」のどちらにもわかりやすいように書かれています。

10日でおぼえる Visual Basic 6.0 実践教室

VBテックラボ&瀬戸 遥 著

SE
SHOEISHA

本書内容に関するお問合せについて

このたびは翔泳社の書籍をお買い上げいただき、誠にありがとうございます。弊社では、読者の皆様からのお問い合わせに適切に対応させていただくため、以下のガイドラインへのご協力をお願い致しております。下記項目をお読みいただき、手順に従ってお問い合わせください。

●ご質問される前に

弊社Webサイトの「Q&Aコーナー」(http://www.shoeisha.com/info/help.asp) をご参照ください。これまで受けたご質問への回答（FAQ）や、的確なご質問方法に関する情報を掲示しています。

●ご質問方法

弊社Webサイトの専用フォームサイト（http://www.shoeisha.com/book/qa/）をご利用ください。記載漏れや独自の用紙等によるご質問、お電話や電子メールによるお問合せ、本書にはさみ込まれたアンケートはがきに記入されたご質問等は、お受けしておりません。

※質問専用シートのお取り寄せについて

Webサイトにアクセスする手段をお持ちでない方は、ご氏名、ご送付先（ご住所／郵便番号／電話番号またはFAX番号／電子メールアドレス）および「質問専用シート送付希望」と明記のうえ、電子メール（qaform@shoeisha.com)、FAX、郵便（80円切手をご同封願います）のいずれかにて“編集部読者サポート係”までお申し込みください。お申し込まれた手段によって、折り返し質問シートをお送りいたします。シートに必要事項を漏れなく記入し、“編集部読者サポート係”までFAXまたは郵便にてご返送ください。

●ご回答について

ご回答は、ご質問いただいた手段によってご返事申し上げます。ご質問の内容によっては、回答に数日ないしはそれ以上の期間を要する場合があります。

●ご質問に際してのご注意

本書の対象を越えるもの、記述個所を特定されないもの、また読者固有の環境に起因するご質問等にはお答えできませんので、予めご了承ください。

●郵便物送付先およびFAX番号

送付先住所　〒160-0006　東京都新宿区舟町5
FAX番号　03-5362-3818
宛先　（株）翔泳社出版局 編集部読者サポート係

※本書に記載されたURL等は予告なく変更される場合があります。
※本書の出版にあたっては正確な記述につとめましたが、著者や出版社などのいずれも、本書の内容に対してなんらかの保証をするものではなく、内容やサンプルに基づくいかなる運用結果に関してもいっさいの責任を負いません。
※本書に掲載されているサンプルプログラムやスクリプト、および実行結果を記した画面イメージなどは、特定の設定に基づいた環境にて再現される一例です。

ようこそ、Visual Basic 6.0実践教室へ！

この全レッスンを受講＆マスターすれば、今日からあなたもフリーソフトプログラマーの仲間入り！

Visual Basicでプログラムを作成するには、基礎的なテクニックに加えて多くのコントロールやプログラミング手法に対する知識とテクニックの習得が必要です。また、より多くのテクニックを身に付けておけば、アイデア次第でさまざまな用途のプログラム開発が可能になっています。

Visual Basicには、とても高度で優秀な機能を持ったコントロールが豊富に用意されています。「こんな機能をプログラムに組み込みたい」と思えば、それに見合ったコントロールを組み込むことで実現します。しかし、それには、コントロールが持っている機能の内容や使い方を知らなければなりません。

今回、「10日でおぼえるVisual Basic 6.0入門教室」の続編として、このコントロールの使い方を中心に、より実践的なテクニックをふんだんに紹介した本書を作成しました。

本書の前半では、リッチテキストボックスやリストビュー、プログレスバーコントロールをはじめとした、便利なActiveXコントロールの使い方を紹介しています。これらのコントロールをプログラムに組み込めば、一層魅力的なソフトに仕上げることができます。更に、1つのフォーム内に複数のウィンドウを開くことができる「MDI（Multi Document Interface)」機能の実装方法や、インターネットのHTTP・FTPサーバとデータの送受信を行うテクニックなども紹介しています。後半では、Win32APIを呼び出す方法を詳しく説明しています。Visual Basicが持っていないWindowsの機能をプログラムに組み込むためには、どうしてもAPIの力を借りなければなりません。本書では、APIの概要説明からコードの実装まで、APIを使い込む上での知識とテクニックを、とてもわかりやすく紹介しています。

また、インターネットエクスプローラとそのコントロールである「WebBrowser」コントロールを使って、自分のプログラムにホームページをブラウザする機能を組み込む方法や、Microsoft Excelを例にとって、Officeアプリケーションをプログラムから操作するテクニックなど、より実践的で幅広いプログラミング技法を紹介しています。本書では、レッスンごとに1つの完成したプログラムを作成するように構成しています。通り一遍の紋切り型でテクニックを解説するのではなく、実際の応用事例に沿ったテクニックの使い方を解説していますので、本書を手にしたその日からすぐにプログラム開発に役立てることができます。

ぜひ、本書を活用して、プログラミングライフをエンジョイしてください。

1999年11月 瀬戸 遥

Let's VBSchool!!

目次

CD-ROMをご使用の前に

CD-ROMを使用する際のご注意等です。CD-ROM中のテキストファイル「readme.txt」もあわせてお読みください。

● CD-ROMについて

巻末に付属のCD-ROMには本書「10日でおぼえるVisual Basic 6.0実践教室」の本文中で解説した各レッスンの完成版やレッスンに必要なファイル（アイコンや画像）が入っています。ハードディスクにフォルダごとコピーしてお使いください。ファイルをコピーする方法がわからないときはWindowsのオンラインヘルプから「ファイルやフォルダをコピーするには」をご覧ください。

※レッスンで使用するプロジェクトファイル（またはHTMLファイル）で、外部ファイルとリンクしているものはリンクの参照情報（フルパス名）が無効になっている場合があります。その際は、必要なファイルをハードディスクにコピーした後、本書の手順に従い自分の環境に合うように再度プロジェクトファイル等を作り直してください。またCD-ROMからコピーしたファイルはすべて［属性］が［読み取り専用］になっており、書き込みが不可になっています。コピーした後は［読み取り専用］のチェックをはずして使用してください。

●免責事項について

CD-ROMに収録されたファイルは通常の運用においては何ら問題ないことを編集部では確認していますが、万一運用の結果、いかなる損害が発生したとしても著者および（株）翔泳社はいかなる責任も負いません。また、収録ファイルに関するテクニカルサポートを行う義務もありません。すべて自己責任においてご使用ください。

● CD-ROMの使用環境条件

レッスンを進めるにはMicrosoft Visual Basic 6.0 Professional Editionが必要です。CD-ROMにはVisual Basicは収録されていません。また「Learning Edition」では特殊なコントロールを使う一部のレッスンが動作しませんので、動作保証はありません。

● CD-ROMのテスト環境について

CD-ROMは以下の環境で正常に動作することを確認しました。

- Microsoft Visual Basic 6.0 (SP3) Professional Edition
- Windows 95/98/Windows NT Workstation 4.0
- Hewlett-Packard Vectra VEi8
- Pentium III 450 MHz

●著作権等について

本書に収録したプログラムおよびソースコードの著作権は、著者および（株）翔泳社が所有します。ただし読者が個人的に利用する場合に関してはソースコードやフォーム、コントロールの流用や改変は自由です。商用利用に関しては、（株）翔泳社へ電子メール(mail-web@shoeisha.co.jp）などでご一報ください。

（株）翔泳社 編集部

Let's VBSchool!!

第1日 (Lesson1 ~ Lesson4)

かんたん表計算ソフトの作成

Visual Basic 6.0に標準で添付されているActiveXコントロールの1つに、「フレキシブルグリッドコントロール」があります。このコントロールは、Excelのようなセルを持ったコントロールで、表計算機能をプログラムに追加したいときなどに使用すると便利です。このコントロールは、セルへのデータの入出力や範囲選択、セルの塗りつぶしやフォント設定などを行うことができます。

第1日のレッスンでは、このフレキシブルグリッドコントロールを使って、簡単な表計算機能を持ったプログラムを作成します。

今日のレッスンで解説する項目

- ❑ プロジェクトへのフレキシブルグリッドコントロールの追加
- ❑ フレキシブルグリッドコントロールの固定セルの設定やセル数の設定など、各プロパティの設定
- ❑ ショートカットメニューの組み込み
- ❑ グラフィック付きコマンドボタンの作成
- ❑ 文字列から整数へのデータ型変換関数
- ❑ 選択したセル番地の把握
- ❑ ランダムアクセスモードに関する知識
- ❑ コモンダイアログボックスの操作
- ❑ データのファイルへの書き出し操作
- ❑ ファイルからデータを読み出す方法
- ❑ エラー処理の設定

Lesson 1

第１日
１時限目

かんたん
表計算ソフト
の作成

Lesson1.vbp
VB6CD2
Lesson01

ユーザーインターフェイスの作成

Visual Basicのプロジェクトに、フレキシブルグリッドコントロールを追加する方法を学習します。また、フレキシブルグリッドコントロールの固定セルの設定やセル数の設定など、各プロパティの設定方法を覚えます。

今回作成するプログラム

1 値を入力し…

簡単表計算ソフト

128900 | Input

2 ここをクリックすると…

	A	B	C	D	合計
1	889000	230000	312000	128900	
2					
3	128900				
4					
5					
6					
7					

3 カレントセルに値が入力される

カレントセルがグレー表示される

POINT

最初に、プログラムの外観を作成します。フレキシブルグリッドコントロールは、セルの行列数などの初期設定を、デザイン時にプロパティウィンドウで行うことができます。

このプログラムは、フォームに６列×10行のセルを持つフレキシブルグリッドコントロールを作成します。そして、テキストボックスコントロールを使ってセルに値を入力するようにします。

コントロールとプロパティの設定

※新しいプロジェクトファイル（標準 EXE）を用意します。また、付録CD-ROMに収録されている「Material」フォルダを、自分のコンピュータのハードディスクにコピーしてください。

1 ツールパレットをマウスの右ボタンでクリックし、メニューから［コンポーネント］を選ぶ

1 ここをクリック

2 ［Microsoft FlexGrid Control 6.0］のチェックボックスをオンにし、［OK］ボタンをクリックする

1 ここをチェック

2 ここをクリック

ツールパレットにフレキシブルグリッドコントロールが追加される

プロジェクトファイルの作成

新しいプロジェクトファイルを作成するには、Visual Basic 6.0を起動し、［新しいプロジェクト］ダイアログボックスで［標準 EXE］をダブルクリックします。

Visual Basic 6.0の基本操作の詳細については、弊社刊『10日でおぼえるVisual Basic 6.0入門教室』を参照してください。

コンポーネント

コンポーネントとは、プログラムのコードを使って、プログラマーが一貫した方法で利用できるプログラム用「部品」の総称です。

フレキシブルグリッドコントロールやコモンダイアログコントロールは、「コントロール」という外部ファイルとして用意されているコンポーネントのひとつの種類です。

フォームを広げる

フォームを広げるには、フォームの周囲に表示されているサイズ変更ハンドル（黒い四角形）を外方向にドラッグします。

プロパティへの文字の入力

プロパティの名前の欄（ここでは［Caption］）をダブルクリックすると、その右側の文字を入力する欄が選択されます。

3 フォームを広げ、［Caption］プロパティを「簡単表計算ソフト」に変更する

プロパティ - Form1

Form1 Form

全体 | 項目別

Appearance	1 - 3D
AutoRedraw	False
BackColor	&H8000000F&
BorderStyle	2 - 可変
Caption	簡単表計算ソフト
ClipControls	True
ControlBox	True
DrawMode	13 - Copy Pen
DrawStyle	0 - 実線 (中心線)
DrawWidth	1

1 「簡単表計算ソフト」と入力

4 ツールパレットの［TextBox］ abl を使って、テキストボックスコントロールを配置する

簡単表計算ソフト

Text1

1 斜めにドラッグして配置

5 ［Text］プロパティの文字を削除する

プロパティ - Text1

Text1 TextBox

全体 | 項目別

OLEDragMode	0 - 手動
OLEDropMode	0 - なし
PasswordChar	
RightToLeft	False
ScrollBars	0 - なし
TabIndex	0
TabStop	True
Tag	
Text	
ToolTipText	
Top	120
Visible	True

1 Delキーを押して削除

6 ツールパレットの［CommandButton］を使って、コマンドボタンを2つ配置する

簡単表計算ソフト

Command1

Command2

1 それぞれ斜めにドラッグして配置

7 ［Command1］ボタンの［Caption］プロパティを「Input」に変更する

プロパティ - Command1

Command1 CommandButton

全体 | 項目別

(ｵﾌﾞｼﾞｪｸﾄ名)	Command1
Appearance	1 - 3D
BackColor	&H8000000F&
Cancel	False
Caption	Input
CausesValidation	True
Default	False
DisabledPicture	(なし)
DownPicture	(なし)
DragIcon	(なし)

1 「Input」を入力

操作の対象を選択する

手順7を行う前に、フォーム上にある［Command1］ボタンをクリックして選択してください。

8 ［Command2］ボタンの［Caption］プロパティの文字を削除する

プロパティ - Command2

Command2 CommandButton

全体 | 項目別

(ｵﾌﾞｼﾞｪｸﾄ名)	Command2
Appearance	1 - 3D
BackColor	&H8000000F&
Cancel	False
Caption	
CausesValidation	True
Default	False
DisabledPicture	(なし)
DownPicture	(なし)
DragIcon	(なし)

1 Delキーを押して削除

操作の対象を選択する

手順8を行う前に、フォーム上にある［Command2］ボタンをクリックして選択してください。

9 [Command2] ボタンの [Picture] プロパティをクリックし、[...] ボタンをクリックする

1 ここをクリック

2 ここをクリック

10 [ピクチャの読み込み] ダイアログボックスで「Check.ico」を選び、[開く] をクリックする

1 「Check.ico」をクリック

2 ここをクリック

11 [Style] プロパティを [1 －グラフィックス] に変更する

1 [1 －グラフィックス] を選択

2 アイコンが表示される

[...] ボタン

[...] ボタンは、そのプロパティの詳細な指定を行うダイアログボックスが用意されていることを示しています。

「Check.ico」ファイル

「Check.ico」ファイルは、付録CD-ROMからコピーした「Material」フォルダに保存されています。[ファイルの場所] ボックスで、「Material」フォルダを選択してください。

選択肢の一覧を表示する

[Style] プロパティの右側の欄の右端にある下向き三角をクリックすると、選択肢の一覧が表示されます。

[Style] プロパティ

このプロパティを [1-グラフィックス] に設定しておくとコマンドボタンに指定したアイコン画像を表示できるようになります。アイコン付きボタンを作るとき設定します。

12 ツールパレットの［MSFlexGrid］ を使って、フレキシブルグリッドコントロールを配置する

簡単表計算ソフト

Input

1 斜めにドラッグして配置

13 ［Cols］プロパティを「6」に設定する

プロパティ - MSFlexGrid1

MSFlexGrid1 MSFlexGrid

全体 | 項目別

BackColorSel	&H8000000D&
BorderStyle	1 - flexBorderSingle
CausesValidation	True
Cols	6
DataSource	
DragIcon	(なし)
DragMode	0 - vbManual
Enabled	True
FillStyle	0 - flexFillSingle
FixedCols	1

1 「6」を入力

14 ［Rows］プロパティを「10」に設定する

プロパティ - MSFlexGrid1

MSFlexGrid1 MSFlexGrid

全体 | 項目別

RightToLeft	False
RowHeightMin	0
Rows	10
ScrollBars	3 - flexScrollBarBo
ScrollTrack	False
SelectionMode	0 - flexSelectionFre
TabIndex	3
TabStop	True
Tag	
TextStyle	0 - flexTextFlat

1 「10」を入力

フレキシブルグリッドコントロール

フレキシブルグリッドコントロールは、行列に複数のセルを持つコントロールです。

15 [ScrollBars] プロパティを [3 − flexScroll BarBoth] に設定する

1 [3 − flexScroll BarBoth] を選択

16 [RowHeightMin] プロパティを「320」に設定する

1 「320」と入力

17 完成したユーザーインターフェイス

[ScrollBars] プロパティ

[ScrollBars] プロパティはフレキシブルグリッドコントロールにスクロールバーを設定するプロパティです。

[ScrollBars] プロパティの右側の欄の右端にある下向き三角をクリックすると、選択肢の一覧が表示されます。

[RowHeight Min] プロパティ

[RowHeightMin] プロパティでは、行全体の高さを「Twip」単位で設定します。

Twip

Visual Basic での画面上の長さを「Twip」という単位で指定します。1論理cmは約567twip、1論理インチは約1,440twipになります。

「1論理cm」とは、印刷したときの長さが1cmになるということです。

コードの記述

Form1:Command1-Click

Lesson1 - Form1 (ｺｰﾄﾞ)

Command1 | Click

```
Private Sub Command1_Click()
    MSFlexGrid1.Text = Text1.Text
    Text1.Text = ""
End Sub
```

Form1:Command2-Click

Lesson1 - Form1 (ｺｰﾄﾞ)

Command2 | Click

```
Private Sub Command2_Click()
    Unload Form1
    End
End Sub
```

Form1:Form-Load

Lesson1 - Form1 (ｺｰﾄﾞ)

Form | Load

```
Private Sub Form_Load()
    Dim i As Integer

    For i = 1 To 4
        With MSFlexGrid1
            .Col = i
            .Row = 0
            .Text = Chr(64 + i)
        End With
    Next

    With MSFlexGrid1
            .Col = 5
            .Row = 0
            .Text = "合計"
        End With

    For i = 1 To 9
        With MSFlexGrid1
            .Col = 0
            .Row = i
```

```
            .Text = i
        End With
    Next

    With MSFlexGrid1
        .Col = 1
        .Row = 1
        .CellBackColor = QBColor(8)
    End With
End Sub
```

Form1:MSFlexGrid1-Click

Lesson1 - Form1 (コード)

MSFlexGrid1 | Click

```
Private Sub MSFlexGrid1_Click()
    MSFlexGrid1.CellBackColor = QBColor(8)

    If MSFlexGrid1.Text <> "" Then
        Text1.Text = MSFlexGrid1.Text
    End If
    Text1.SetFocus
End Sub
```

Form1:MSFlexGrid1-LeaveCell

Lesson1 - Form1 (コード)

MSFlexGrid1 | LeaveCell

```
Private Sub MSFlexGrid1_LeaveCell()
    MSFlexGrid1.CellBackColor = 0
End Sub
```

解　説

1　フレキシブルグリッドコントロールは、行列に複数のセルを持つコントロールです。グレーに塗られたセルが「固定セル」と呼ばれるセルで、行列の見出しなどに使われるセルです。フォームデザイン時は選択したり値を代入するなどの操作はできません。

白く表示されているセルが、実際の値を表示するセルで、フォームに配置した時点では1つしか配置されていません。

簡単表計算ソフト

Input

フォーム配置時は
行列２列

2　行列数を設定するのが、［Cols］と［Rows］プロパティです。これらは、固定セルを含めた総行列数を設定します。このプログラムでは10行6列の表を作成しています。

固定セルは、行列ともに1つずつ設定されていますが、［FixedCols］と［FixedRows］プロパティを使用すれば、複数の固定セルを設定できます。

簡単表計算ソフト

Input

複数の固定セルを
設定

3　［ScrollBars］プロパティは、フレキシブルグリッドコントロールにスクロールバーを設定するかどうかを指定するプロパティです。このプロパティを設定しておけば、フレキシブルグリッドコントロールのサイズが、設定した行列数よりも小さい場合（つまり、すべてのセルが表示しきれない場合）に、自動的にスクロールバーが表示されます。

［RowHeightMin］プロパティは、行全体の高さを「Twip」単位で設定します。また、［RowHeight］プロパティを使うと、指定した行だけ高さを変更できます。列の幅を変更するには、［ColWidth］プロパティに幅をTwip単位で指定します。ただし、このプロパティは一列単位で設定します。

［AllowUserResizing］プロパティを使用すると、プログラム実行時にユーザーがセルの幅や高さを自由に変更できるようになります。

ユーザーがセルの幅や高さを変更できるようになる

4　また、Font関係のプロパティを操作すると、セルに表示されるデータのフォント属性を設定できます。これらの、フレキシブルグリッドコントロールの初期設定は、プロパティページを使って設定すると便利です。プロパティウィンドウにある［(プロパティページ)］を選び、［...］ボタンをクリックすると、フレキシブルグリッドコントロールの［プロパティページ］ダイアログボックスが表示され、初期値を一括して設定することができます。

プロパティウィンドウの［(プロパティページ)］をクリックし、［...］をクリック

［プロパティページ］ダイアログボックス

5 コマンドボタンコントロールは、［Style］プロパティを［グラフィックス］に設定し、［Picture］プロパティにアイコンやビットマップを割り当てると、グラフィカルなコマンドボタンを作成できます。また、［Caption］プロパティに文字を設定すれば、文字とグラフィックを持ったコマンドボタンを作成できます。

グラフィックの付いたコマンドボタン

グラフィックと文字の付いたコマンドボタン

6 では、コードの説明を行います。このプログラムは、セルをクリックして、テキストボックスコントロールに数値を入力し、［Input］ボタンをクリックするとフレキシブルグリッドコントロールのセルにその値を代入します。

フレキシブルグリッドコントロールは、実行時に値を直接セルに入力することができません。したがって、他の入力用インターフェイスを用意し、そこからコードで値をセルに転送する必要があります。このプログラムでは、テキストボックスコントロールを入力用インターフェイスにしています。

1 セルをクリック
2 値を入力
3 ここをクリック
4 値が入力される

7

最初に、フォームのLoadイベントプロシージャで、フレキシブルグリッドコントロールの初期設定を行います。

まず、For...Nextループを使って、列方向の固定セルに項目名を入力します。固定セルへの入力は、プログラムデザイン時にプロパティウィンドウで行うことができないので、コードで行います。

セルへの値の代入は、まず入力するセル番地を指定してカレントセルにします。そして、［Text］プロパティに文字を代入します。カレントセルの指定は、フレキシブルグリッドコントロールをオブジェクトに指定し、［Col］と［Row］プロパティに数字を設定します（セル番地は、固定セルの左上が「0,0」になります）。

たとえば、

```
MSFlexGrid1.Col = 1
MSFlexGrid1.Row = 0
MSFlexGrid1.Text = "ABC"
```

というコードを実行すると、固定セルの左から2番目に「ABC」という文字が入力されます。

簡単表計算ソフト

ABC

「ABC」と入力される

8

これを応用して、For...Nextループを使って固定セルの列方向に、自動的に「A」「B」「C」と英字を振っていきます。文字を自動入力するには、いちいち文字を指定するよりもChr()関数を使用します。

この関数は、ASCII文字セットをコード番号で指定すると、文字に変換して返してきます。たとえば、「Str=Chr(65)」と実行すると、変数「Str」には文字「A」が格納されます。この関数を使って、文字コードにループのカウンタ用変数（i）を加えて、固定セルの1列目に「A」、2列目に「B」、3列目に「C」と順番に英字を入力していきます。

```
Private Sub Form_Load()
    Dim i As Integer

    For i = 1 To 4
        With MSFlexGrid1
            .Col = i
            .Row = 0
            .Text = Chr(64 + i)
        End With
    Next
```

9 セルの一番右端の列は、合計を入力する欄にするので、固定セルには「合計」の文字列を入力します。これは、直接このセルをカレントセルに指定し、[Text] プロパティに文字列を代入します。

```
With MSFlexGrid1
    .Col = 5
    .Row = 0
    .Text = "合計"
End With
```

また、列見出しと同じ方法で、固定行に見出しを付けます。今度は数字なので、[Row] プロパティと [Text] プロパティにループのカウンタ用変数を割り当て、上から順番に1行目に「1」、2行目に「2」…と番号を付けていきます。

```
For i = 1 To 9
    With MSFlexGrid1
        .Col = 0
        .Row = i
        .Text = i
    End With
Next
```

10

最後に、セルの背景色を灰色に塗りつぶします。これは、現在どのセルがカレントセルなのかがわかるように、セルの背景色を塗りつぶすことで識別します。

最初に、[Col] と [Row] プロパティに1を代入して、データ入力可能なセルのもっとも左上のセルをカレントセルにします。

次に、そのセルの背景を塗りつぶします。セルの背景色の塗りつぶしには、[CellBackColor] プロパティを使用します。プロパティの値には色の番号を設定しますが、これらはQBColor関数またはRGB関数を使用して、その戻り値をプロパティに設定します。このプログラムでは、QBColor関数を使用しました。

```
With MSFlexGrid1
    .Col = 1
    .Row = 1
    .CellBackColor = QBColor(8)
End With
```

11

フォーム起動時の処理が終わったら、今度は [Input] ボタンの処理を行います。テキストボックスコントロールに入力された文字をセルに代入します。

これは、テキストボックスコントロールの [Text] プロパティの値を、そのままフレキシブルグリッドコントロールの [Text] プロパティに代入するだけです。代入したら、次の入力に備えてテキストボックスコントロールの [Text] プロパティの値を空にしておきます。

```
Private Sub Command1_Click()
    MSFlexGrid1.Text = Text1.Text
    Text1.Text = ""
End Sub
```

フレキシブルグリッドコントロールの［Text］プロパティに文字が代入されると、現在カレントになっているセルにその値が表示されます。ですから、好きなセルをクリックしてカレントセルにしてから［Input］ボタンをクリックすれば、そのセルに値が表示されます。

簡単表計算ソフト

12345　Input

	A	B	C	D	合計
1					
2					
3					
4					
5					
6					
7					

セルを指定し、値を入力して［Input］ボタンをクリックすると...

簡単表計算ソフト

Input

	A	B	C	D	合計
1					
2					
3			12345		
4					
5					
6					
7					

そのセルに値が転送される

12　今度は、ユーザーがセルをクリックした場合の処理を作成します。ユーザーが、現在のセルから他のセルをクリックすると、フレキシブルグリッドコントロールには「Click」イベントが発生します。

最初にカレントセルの背景色を塗りつぶしていますから、ユーザーのクリックによってカレントセルが移動した場合は、今度はそのセルの背景を塗りつぶすようにします。

セルがクリックされると、自動的にフレキシブルグリッドコントロールのClickイベントプロシージャが呼び出されますから、このプロシージャをコードウィンドウのイベントリストボックスを使って表示します。

Click イベントでは、最初にクリックされたセルの背景色を塗りつぶします。クリックされた時点で、このセルがカレントセルになりますから、そのまま［CellBackColor］プロパティに「グレー」を設定します。

```
Private Sub MSFlexGrid1_Click()
    MSFlexGrid1.CellBackColor = QBColor(8)
```

13 次に、If ステートメントを使用して、クリックしたセルに値が格納されているかどうかを判断します。もし値が格納されていれば、修正できるようにその値をテキストボックスコントロールに転送します。

値があるかどうかの判断には、「<>」演算子を用いて、フレキシブルグリッドコントロールの［Text］プロパティが空でないことを確認します。［Text］プロパティが空でなければ（つまり、値が格納されていれば）、その値をテキストボックスコントロールに表示します。空であれば、何もせずにそのまま［Input］ボタンの操作が実行されます。

```
If MSFlexGrid1.Text <> "" Then
    Text1.Text = MSFlexGrid1.Text
End If
```

そして、次の入力に備えて、テキストボックスコントロールにフォーカスを移しておきます。

```
Text1.SetFocus
```

14　今度は、クリックされたときに、それまでカレントだったセルの処理を作成します。それまでカレントだったセルは、他のセルがクリックされるとカレントセルでなくなりますから、セルの背景色を元に戻しておく必要があります。

　フレキシブルグリッドコントロールでは、カレントセルが他のセルに変更される直前に、「LeaveCell」イベントが発生します。このイベントプロシージャに、カレントだったセルの背景色を戻す処理を作成します。

　コードウィンドウのイベントリストボックスから、「LeaveCell」イベントを選び、このイベントプロシージャを作成します。そして、［CellBackColor］プロパティの値を「0」に設定します。プロパティの値が「0」になると、セルの背景色は既定値に設定されます。

　これで、前述の「MSFlexGrid1_Click」プロシージャの処理とあわせ、ユーザーがセルをクリックするとセルの背景色が入れ替わり、カレントセルが移動したことがユーザーの目にもわかるようになります。

```
Private Sub MSFlexGrid1_LeaveCell()
    MSFlexGrid1.CellBackColor = 0
End Sub
```

15　最後に、［終了］ボタンの処理を作成します。これは、もうすっかりおなじみになった、フォームのアンロードとEndステートメントによるプログラムの終了処理を行います。

```
Private Sub Command2_Click()
    Unload Form1
    End
End Sub
```

16　以上で、プログラムの基本的な入力処理が完成です。プログラムを実行させ、テキストボックスコントロールに数値や文字を入力し、［Input］ボタンをクリックすると、カレントセルに値が転送されます。また、セルをクリックするとグレーに塗りつぶされ、入力されるセルがよくわかるようになります。

! ワンポイントアドバイス

フレキシブルグリッドに用意されたイベントは「LeaveCell」以外にもたくさんあります。イベントをうまく利用していくことがプログラミングのコツです。

<フレキシブルコントロールに用意されているイベント>

Click, Compare, DblClick, DragDrop, DropOver, EnterCell, GotFocus, KeyDown, KeyPress, KeyUp, LeaveCell, LostFocus, MouseDown, MouseMove, MouseUp, OLECompleteDrag, OLEDragDrop, OLEDragOver, OLEGiveFeedback, OLESetData, OLEStartDrag, RowColChange, Scroll, SelChange

まとめ

このレッスンでは、フレキシブルグリッドコントロールの初期設定を行う方法を、各プロパティの機能とあわせて説明しました。また、セルにデータを入力するための基本的な処理を説明しました。フレキシブルグリッドコントロールは、豊富なプロパティを持っており、いろいろな設定が行えるようになっています。

Excelのように直接セルに値を入力できませんので、他のコントロールと組み合わせてデータの入力を行うようにするのがポイントです。

ワンポイントアドバイス

[CellBackColor] プロパティの色番号の指定には、2通りの方法があります。

【QBColor関数の場合】

```
CellBackColor = QBColor(8)
```

【RGB関数の場合】

```
CellBackColor = RGB(255, 245, 112)
```

QBColor関数は、あらかじめ色が数値で設定されているので、数値と色がすぐにわかります。ただし、16色までしか指定できません。

一方、RGB関数は、光の3原色である「Red」「Green」「Blue」の3つの色を表す数値を「0～255」の範囲で指定して、色番号を導き出します。この関数を使うとRGBの組み合わせで1670万色中の1色を選べますから、好きな色を使うことができます。

＜QBColor関数にセットする色番号＞

番号	色	番号	色
0	黒	8	灰色
1	青	9	明るい青
2	緑	10	明るい緑
3	シアン	11	明るいシアン
4	赤	12	明るい赤
5	マゼンタ	13	明るいマゼンタ
6	黄	14	明るい黄
7	白	15	明るい白

Q 練習問題

Q1 作成した「Lesson1.vbp」のフレキシブルグリッドコントロールのセルの幅と高さが、ユーザーによって自由に調節、変更できるようにしてください。

解答は巻末に→

COLUMN ・・・・・

演算子の種類と優先順位

Visual Basicで用いられる演算子には「算術演算子」、「比較演算子」、「論理演算子」の3種類があります。P.18の「13」で説明している「<>」演算子は、比較演算子と呼ばれるものです。

算術演算子	比較演算子	論理演算子
指数演算（^）	等しい (=)	Not
マイナス符号（-）	等しくない（<>）	And
乗算と除算（*、/）	より小さい（<）	Or
整数除算（¥）	より大きい（>）	Xor
剰余演算（Mod）	以下（<=）	Eqv
加算と減算（+、-）	以上（>=）	Imp
文字列連結（&）	Like	
	Is	

式の中で複数の演算が行われるときは、「算術演算子」→「比較演算子」→「論理演算子」の順番に行われていきます。また算術演算子の中ではこの表の上から下の順番に評価されていきます。例えば文字列連結演算子「&」は比較演算子より前に評価されますが、算術演算子の中では一番最後に評価されます。

Lesson 2

第１日
２時限目

かんたん
表計算ソフト
の作成

Lesson2.vbp
VB6CD2
Lesson02

合計計算処理の作成

作成したプログラムに、セルの数値を合計する処理を組み込みます。複数のセルを選択して、［合計］というメニューを選ぶと、選択したセルの値を合計する計算を実行します。メニューは、ポップアップメニューを採用し、選択範囲の位置で選べるようにします。

今回作成するプログラム

1 合計したいセルを左から右に選択する

簡単表計算ソフト

	A	B	C	D	合計
1	1000	2000	3000	4000	10000
2					
3					
4					
5					
6					
7					

2 右クリックしてポップアップメニューの［合計］をクリックすると合計が表示される

POINT

プログラムにポップアップメニューを組み込み、列方向に選択したセルのデータを合計計算します。範囲選択したセル番地の把握と値の取り出し方法、計算式の組み立て方法の知識を身に付けます。

コントロールとプロパティの設定

※前のレッスンで作成した「Lesson1.vbp」を開いてください。

1 ［標準］ツールバーの［メニューエディタ］ボタンをクリックし、メニューエディタを起動する

メニュー エディタ
キャプション(P):
名前(M):
インデックス(X):
ショートカット(S): (なし)
ヘルプ コンテキスト番号(H): 0
ネゴシエート位置(O): 0 - なし
チェック(C)　有効(E)　表示(V)　ウィンドウ リスト(W)
OK　キャンセル
次へ(N)　挿入(I)　削除(T)

［メニューエディタ］ダイアログボックス

2 ［キャプション］に「計算」、［名前］に「IDM_CALC」と入力する

メニュー エディタ
キャプション(P): 計算
名前(M): IDM_CALC

1 「計算」と入力

2 「IDM_CALC」と入力

3 ［表示］チェックボックスをクリックしてオフにする

メニュー エディタ
キャプション(P): 計算
名前(M): IDM_CALC

1 ここをクリック

Lesson1.vbp ファイル

「Lesson1.vbp」ファイルは、付録CD-ROMの「Lesson01」フォルダにも収録されています。

［表示］チェックボックス

トップメニューの［表示］チェックボックスをオフにすると、その下に組み込まれるサブメニューも含め、そのメニュー全体が非表示になります。

ポップアップメニューを作るときはこのようにチェックを外しておきます。

メニューの挿入

「IDM_CALC」を入力したところでEnterキーを押しても、同様に空白のメニューを挿入することができます。

4 [次へ] をクリックして次の行に移動する

メニュー エディタ

キャプション(P):	計算		OK
名前(M):	IDM_CALC		キャンセル
インデックス(X):		ショートカット(S):	(なし)
ヘルプ コンテキスト番号(H):	0	ネゴシエート位置(O):	0 - なし

□ チェック(C)　☑ 有効(E)　□ 表示(V)　□ ウィンドウ リスト(W)

← → ↑ ↓ 次へ(N) 挿入(I) 削除(T)

計算

1 ここをクリック

5 次の行で右向きの矢印ボタンをクリックして一段下がったメニューを挿入する

メニュー エディタ

キャプション(P):			OK
名前(M):			キャンセル
インデックス(X):		ショートカット(S):	(なし)
ヘルプ コンテキスト番号(H):	0	ネゴシエート位置(O):	0 - なし

□ チェック(C)　☑ 有効(E)　☑ 表示(V)　□ ウィンドウ リスト(W)

← → ↑ ↓ 次へ(N) 挿入(I) 削除(T)

計算
····

1 ここをクリック

6 [キャプション] に「合計」、[名前] に「IDM_SUM」と入力する

メニュー エディタ

キャプション(P):	合計		OK
名前(M):	IDM_SUM		キャンセル
インデックス(X):		ショートカット(S):	(なし)
ヘルプ コンテキスト番号(H):	0	ネゴシエート位置(O):	0 - なし

□ チェック(C)　☑ 有効(E)　☑ 表示(V)　□ ウィンドウ リスト(W)

← → ↑ ↓ 次へ(N) 挿入(I) 削除(T)

計算
····合計

1 「合計」と入力

2 「IDM_SUM」と入力

7 ［OK］ボタンをクリックしてメニューエディタを閉じる

メニュー エディタ

キャプション(P):	合計
名前(M):	IDM_SUM
インデックス(X):	
ショートカット(S):	(なし)
ヘルプ コンテキスト番号(H):	0
ネゴシエート位置(O):	0 - なし

☐ チェック(C)　☑ 有効(E)　☑ 表示(V)　☐ ウィンドウ リスト(W)

OK　キャンセル　次へ(N)　挿入(I)　削除(T)

計算
····合計

1 ［OK］ボタンをクリック

コードの記述

Form1:MSFlexGrid1-MouseDown

Lesson2 - Form1 (コード)

MSFlexGrid1　MouseDown

```
Private Sub MSFlexGrid1_MouseDown(Button As Integer, Shift As _
    Integer, x As Single, y As Single)
    If Button = 2 Then
        PopupMenu IDM_Calc
    End If
End Sub
```

Form1:IDM_SUM-Click

Lesson2 - Form1 (コード)

IDM_SUM　Click

```
Private Sub IDM_Sum_Click()
    Dim StartCol As Integer
    Dim EndCol As Integer
    Dim Store As Long

    Store = 0
    With MSFlexGrid1
        StartCol = .Col
        EndCol = .ColSel
    End With

    On Error GoTo BACK
```

```
    For i = StartCol To EndCol
        With MSFlexGrid1
            .Col = i
            Store = Store + CLng(.Text)
        End With
    Next

    With MSFlexGrid1
        .Col = 5
        .Text = Store
    End With
    Exit Sub

BACK:
    MsgBox "入力データが数値ではないので計算できません。"

End Sub
```

解説

1 この合計計算の機能は、いろいろな方法が考えられますが、ここでは整数の足し算を行う機能に絞って、次の動作で実現するようにしています。

①列方向に計算したいセルを範囲選択する

②その上でマウスの右ボタンを押すと［合計］メニューがポップアップする

③メニューを選ぶと「合計」欄に合計が入力される

④セルに数値データ以外の値が入力されている場合は、計算できない旨のメッセージを表示する

2 まず、メニューの［合計］を表示する処理を作成します。フレキシブルグリッドコントロールの上でマウスの右ボタンを押すと、「MouseDown」というイベントが発生しますから、このイベントプロシージャに、［合計］メニューを表示する処理を作成します。このプロシージャでは、まずIfステートメントでマウスの右ボタンが押されたかどうかを判断し、押されていれば「PopupMenu」メソッドを使って、［計算］メニューを表示します。

メニューエディタで［計算］メニューを作成したときに、このメニューを「非表示」にしていますから、フォームにはメニューが表示されません。しかし、［表示］メニューの中のサブメニュー［合計］は、「PopupMenu」メソッドを使うと表示されるようになります。

```
Private Sub MSFlexGrid1_MouseDown(Button As Integer, _
    Shift As Integer, x As Single, y As Single)
    If Button = 2 Then
        PopupMenu IDM_CALC
    End If
End Sub
```

3 合計計算の処理は、この［合計］メニューが選ばれた時点で実行しますので、このメニューのClickイベントプロシージャに処理を作成します。
まず、変数を3つ作成します。それぞれの変数は次の役割をします。

```
Dim StartCol As Integer ----- ①
Dim EndCol As Integer -------- ②
Dim Store As Long ------------ ③
```

①選択されたセルの最初のセル（選択範囲の先頭セル）の列番号を格納する変数
②選択されたセルの最後のセル（選択範囲の末尾セル）の列番号を格納する変数
③合計計算値を格納する変数

このとき、変数「Store」だけデータ型を「Long」（長整数型）で宣言します（「Integer」（整数型）では、格納できる整数の桁数が32,767までしかありません）。
変数を宣言したら、変数「Store」に「0」を代入して初期化しておきます。

```
Store = 0
```

4 次に、現在選択されているセルの範囲を把握します。選択範囲の先頭のセルは、現在カレントになっているセルなので、［Col］プロパティを使って列番号を変数「StartCol」に格納します。また、選択範囲の最後のセルは、［ColSel］プロパティを参照することで把握できます（フレキシブルグリッドコントロールの選択範囲は、列方向では［Col］プロパティと［ColSel］プロパティで指定された範囲になります）。
これで、列方向に選択されたセル範囲を把握できます。

```
With MSFlexGrid1
    StartCol = .Col
    EndCol = .ColSel
End With
```

! ワンポイントアドバイス

行方向の選択範囲を把握するには、［Row］プロパティと［RowSel］プロパティを使用します。また、コードでこれらのプロパティに値を設定すれば、コードから範囲を選択することができます。デザイン時には、値を設定できません。

5

選択範囲が把握できたら、セルから値を取り出して合計計算を実行します。ただし、セルに数値以外の値（文字列など）が入力された場合は計算がエラーになるので、合計計算の前に「On Error」ステートメントでエラー処理を挿入しておきます。

合計計算はまず、For...Nextループで、選択範囲の先頭のセルから、範囲の最後のセルまでを１つずつアクセスし、［Text］プロパティの値を取り出します。取り出した値は「CLng」関数を使って、Long型の整数に変換します。変換した値は変数「Store」に加算しながら格納します。これで、範囲の最後のセルまで、［Text］プロパティの値が整数に変換されて合計計算されていきます。

```
On Error GoTo BACK
For i = StartCol To EndCol
    With MSFlexGrid1
        .Col = i
        Store = Store + CLng(.Text)
    End With
Next
```

6

合計計算が終了したら、その値を「合計」欄のセルに格納します。そして、エラー処理のために「Exit（Sub)」ステートメントでプロシージャを終了させます。このステートメントを記述しておかないと、エラーでもないのに、次のエラー処理のコードまで実行されてしまいます。

```
With MSFlexGrid1
    .Col = 5
    .Text = Store
End With
Exit Sub
```

7

CLng関数の引数に数字以外の文字列が入ると、正常な変換ができなくなり、この関数はエラーを発生させます。これをOn Errorステートメントで把握して、次のエラー処理である「BACK」ラベルに処理を移動させます。「BACK」ラベルでは、メッセージボックスで計算できない旨の警告を表示し、プロシージャを終了します。

```
BACK:
    MsgBox "入力データが数値ではないので計算できません。"
```

以上で、合計計算機能が完成です。実際に数値を入力して、計算させてみてください。

まとめ

レッスンの冒頭にも言いましたが、合計計算機能は、他の方法でも実現できます。たとえば、コマンドボタンを使ったり、フォームのメニューを使用する方法もあります。また、合計だけでなく、いろいろな計算を選べるような機能を組み込むこともできます。このレッスンを参考に、自分の設計にあった機能を組み込んでください。

練習問題

Q1 作成したプログラムを、小数点の合計計算もできるように、計算式を組み替えてください。

解答は巻末に→

ワンポイントアドバイス

データ変換関数は、CLng 以外にも関数が用意されています。これらの関数は、引数 x に変換元のデータを与えると、指定した変換操作を行って、その値を返してきます。

＜代表的な変換関数＞

関数名（x は引数）	機能
CBool(x)	ブール型 (Boolean)
CByte(x)	バイト型 (Byte)
CCur(x)	通貨型 (Currency)
CDate(x)	日付型 (Date)
CDbl(x)	倍精度浮動小数点数型 (Double)
CDec(x)	10 進型 (Decimal)
CInt(x)	整数型 (Integer)
CLng(x)	長整数型 (Long)
CSng(x)	単精度浮動小数点数型 (Single)
CVar(x)	バリアント型 (Variant)
CStr(x)	文字列型 (String)

ワンポイントアドバイス

このプログラムでは、合計するセルを選択するときは必ず「左から右」へ選択して、合計計算を行うように作成しています。

「右から左」に選択した場合でも合計計算が行えるようにするには、右から左へ選択されたときと逆の処理を行うようなコードを追加してあげます。

まず、選択範囲の先頭のセル番号と終端のセル番号を比較することで、どちらの方向から選択されたのかがわかりますから、これをIfステートメントで判断し、右から左にセル内容を合計計算する場合と、左から右に合計計算する場合の、２種類の処理に振り分けるようにします（修正したプログラムは、付録CD-ROMに「Lesson2-2.vbp」というファイル名で収録してあります）。

```
On Error GoTo BACK
If StartCol < EndCol Then                    '追加したコード
     For i = StartCol To EndCol
          With MSFlexGrid1
               .Col = i
               Store = Store + CLng(.Text)
          End With
     Next
'右から左へ選択されたときは逆の処理を行えばよい
Else                                          '追加したコード
     For i = EndCol To StartCol               '変更したコード
          With MSFlexGrid1                    '変更したコード
               .Col = i                       '変更したコード
               Store = Store + CLng(.Text)    '変更したコード
          End With                            '変更したコード
     Next
End If                                        '追加したコード
```

Lesson 3

第１日
３時限目

かんたん
表計算ソフト
の作成

Lesson3.xls
VB6CD2
Lesson03

セルデータをファイルに保存する

作成した表計算プログラムのセルデータを、ファイルに保存します。ファイルの読み書きは、ランダムアクセスモードで行います。また、ファイル名の設定には、コモンダイアログボックスを使用します。

● 今回作成するプログラム

［保存］コマンドで［データを保存］ダイアログボックスを表示し、セルデータをファイルに保存する

POINT

ランダムアクセスモードは、データ１つ１つにレコード番号を付けてファイルに書き込むため、任意のデータを抜き出すことができます。この機能を使用して、先頭のセルからセルデータのファイルへの書き出しを行います。コモンダイアログボックスの［ファイル名を付けて保存］ダイアログボックスを表示し、保存するファイル名とフォルダ名を取得します。

コントロールとプロパティの設定

※前のレッスンで作成した「Lesson2.vbp」を開いてください。

1 ツールパレットをマウスの右ボタンでクリックし、メニューから［コンポーネント］を選ぶ

コンポーネント(O)...
タブの追加(A)...
ドッキング可能(K)
非表示(H)

1 ここをクリック

2 ［Microsoft Common Dialog Control 6.0］をチェックし、［OK］ボタンをクリックする

コンポーネント
コントロール　デザイナ　挿入可能なオブジェクト
Microsoft Agent Control 2.0
Microsoft Calendar Control 9.0
Microsoft Chart Control 6.0 (OLEDB)
Microsoft Comm Control 6.0
Microsoft Common Dialog Control 6.0
Microsoft Data Bound Grid Control 5.0 (SP3)
Microsoft Data Bound List Controls 6.0
Microsoft DataGrid Control 6.0 (OLEDB)
Microsoft DataList Controls 6.0 (OLEDB)
Microsoft DataRepeater Control 6.0 (OLEDB)
参照(B)...
選択された項目のみ(S)
Microsoft Common Dialog Control 6.0
場所:　C:¥WINDOWS¥SYSTEM¥COMDLG32.OCX
OK　キャンセル　適用(A)

1 ここをチェック

2 ここをクリック

CommonDialog

ツールパレットにコモンダイアログコントロールが追加される

Lesson2.vbp ファイル

「Lesson2.vbp」ファイルは、付録CD-ROMの「Lesson02」フォルダにも収録されています。

3 ツールパレットの［CommonDialog］を使って、コモンダイアログボックスコントロールを配置する

4 プロパティウィンドウの［(プロパティページ)］をクリックし、［...］ボタンをクリックする

［...］ボタン

［...］ボタンは、そのプロパティの詳細な指定を行うダイアログボックスが用意されていることを示しています。

5 ［開く/保存］タブで、次の項目を設定する

項目	値
初期ディレクトリ	c:¥
フィルタ	データファイル (*.eca)¦*.eca
既定の拡張子	eca
キャンセル時にエラー	(チェックする)

6 [標準] ツールバーの [メニューエディタ] ボタンをクリックし、メニューエディタを起動する

メニュー エディタ
キャプション(P): 計算　OK
名前(M): IDM_CALC　キャンセル
インデックス(X):　ショートカット(S): (なし)
ヘルプ コンテキスト番号(H): 0　ネゴシエート位置(O): 0 - なし
チェック(C)　有効(E)　表示(V)　ウィンドウ リスト(W)
次へ(N)　挿入(I)　削除(T)
計算
・・・・合計

7 [挿入] ボタンをクリックし、メニューを追加する

メニュー エディタ
キャプション(P):　OK
名前(M):　キャンセル
インデックス(X):　ショートカット(S): (なし)
ヘルプ コンテキスト番号(H): 0　ネゴシエート位置(O): 0 - なし
チェック(C)　有効(E)　表示(V)　ウィンドウ リスト(W)
次へ(N)　挿入(I)　削除(T)
計算
・・・・合計

1 [挿入] ボタンをクリック

フォームを選択する

手順6を行う前に、フォーム上のコントロール以外の部分をクリックして、フォームを選択してください。

8 [キャプション] に「ファイル」、[名前] に「IDM_FILE」と入力する

1 「ファイル」と入力

2 「IDM_FILE」と入力

9 さらにメニューを挿入し [キャプション] と [名前] に次の項目を入力し、2つのサブメニューを追加する

[キャプション]	[名前]
開く	IDM_OPEN
保存	IDM_SAVE

10 完成したフォームとメニュー

サブメニューを作成する

サブメニューを作成するには、[挿入] ボタンをクリックして空白行を作り、サブメニューを作成する行を選択して、右向きの矢印ボタンをクリックします。

コードの記述

Form1:IDM_SAVE-Click

Lesson3 - Form1 (コード)

IDM_SAVE　Click

```
Private Sub IDM_SAVE_Click()
    Dim GetFileName As String
    Dim Data As String * 10
    Dim DataByte As Long
    Dim RecNo As Integer
    Dim i As Integer, j As Integer

    RecNo = 1
    With CommonDialog1
        .DialogTitle = "データの保存"
        On Error GoTo BACK
        .ShowSave
    End With

    GetFileName = CommonDialog1.filename
    DataByte = LenB(Data)

    Open GetFileName For Random As #1 Len = DataByte

    For i = 1 To 9
        For j = 1 To 5
            With MSFlexGrid1
                .Col = j
                .Row = i
                Data = .Text
            End With
            Put #1, RecNo, Data
            RecNo = RecNo + 1
        Next
    Next

    Close #1
    Exit Sub

BACK:

End Sub
```

解 説

1 この機能は、セルに入力されたデータを、テキスト形式のランダムアクセスモードでファイルに保存します。メニューの［保存］を選ぶと、コモンダイアログボックスコントロールの［ファイル名を付けて保存］が開きます。ここに、保存するファイル名とフォルダ名を指定し、Openステートメントを実行します。

Openステートメントとは、ファイルを開いて、ファイルへデータを入出力するためのステートメントです。

【Openステートメントの書式】

```
Open pathname For mode [Access access] [lock] As
   [#]filenumber [Len=reclenthe]
```

pathname	開くファイルのパス名を指定します。
mode	ファイルのモードを指定します。Append、Binary、Input、Output、Random。
Access	開くファイルに対する処理を指示します。省略可。
lock	他のプロセスからのアクセスを制御します。省略可。
filenumber	開くファイルに対して参照のための任意のファイル番号を指定します。
reclength	レコード長やバッファ容量を示します。32,787バイト以下の数字を指定します。省略可。

「ランダムアクセスモード」とは、「シーケンシャルアクセスモード」のように、ファイルの先頭から順番にデータを書き込んだり読み出すのではなく、データ一件一件に「レコード番号」を設定して、ファイル内の任意の位置のデータを読み出したり、任意の位置に書き込むことのできるモードです。

また、ランダムアクセスモードは、1つのデータあたりのバイトサイズを設定して書き込みます。このデータサイズは、1つのファイルですべてのデータが同じサイズになります。したがって、設定したバイトサイズに達しないデータでも、その分の領域を消費します。

たとえば、1つのデータのサイズを20バイトに設定した場合、1バイトのデータでも20バイトの領域を消費します。

2 コモンダイアログボックスの各プロパティの初期設定は、プロパティウィンドウでも設定しますが、多くの項目を変更するときは「プロパティページ」を使用したほうが一括して設定できるので便利です。

今回設定した項目とプロパティ名と値は、次のとおりです。

● ［初期ディレクトリ］（InitDir）…「c:¥」

ダイアログボックスが表示されたときに表示される、ディレクトリ（フォルダ）

名を設定します。ここでは、ドライブCのルートディレクトリを指定しています。

● [フィルタ]（Filter）…「データファイル (*.eca)|*.eca」

ファイルのリストを表示するときに、特定の種類のファイルだけを表示したい場合に、このプロパティを設定します。値の書式は、次のとおりです。

[表示タイトル | ファイルの拡張子]

最初にボックスに表示する文字列を指定します。ここでは、「データファイル(*.eca)」を指定しています。次に、「|」（半角縦棒）で区切って、ファイルの種類を表すファイル拡張子を指定します。ここでは、「.eca」を設定しています。必ず「.」（ピリオド）と拡張子の文字（3文字以内）を記述します。このフィルタは、「|」（縦棒）で区切ると、いくつでも設定でき、ダイアログボックスのリストボックスに表示されます。

● [既定の拡張子]（DefaultExt）…「eca」

このプロパティに拡張子を設定しておくと、ファイルを保存する際に、ファイル名を入力するだけで自動的にこの拡張子を付けて保存してくれます。このプロパティを設定していない場合は、ユーザーが拡張子まで付けてファイル名を入力する必要があります。

● [キャンセル時にエラー]（CancelError）…（チェック）

[ファイル名を付けて保存] ダイアログボックスの [キャンセル] ボタンがクリックされたときに、エラーを発生させます。[キャンセル] ボタンがクリックされたときの処理を組み立てる場合は、このプロパティを「True」に設定します。

プロパティ ページ

開く/保存 | 色 | フォント | 印刷 | ヘルプ

タイトル(T):
ファイル名(I):
初期ディレクトリ(D): c:¥
フィルタ(R): データファイル (*.eca)|*.eca
フラグ(F): 0
既定の拡張子(U): eca
最大ファイル サイズ(M): 260
フィルタ インデックス(N): 0
☑ キャンセル時にエラー(E)

OK　キャンセル　適用(A)　ヘルプ

ワンポイントアドバイス

「eca」という拡張子は、このプログラムのために筆者が勝手に作成した拡張子で、「Easy Calc」の略です。ファイルの拡張子は、自由に設定してかまいませんが、現在すでにあるファイルの拡張子と重複しないようにしてください。

3 では、最初にデータをファイルに保存する処理から作成します。この処理は、メニューの［ファイル－保存］を選んだときに実行しますので、このメニューのClickイベントプロシージャに作成します。

プロシージャの先頭で、使用する変数を宣言します。

```
Dim GetFileName As String            ----------- ①
Dim Data As String * 10              ----------- ②
Dim DataByte As Long                 ----------- ③
Dim RecNo As Integer                 ----------- ④
Dim i As Integer, j As Integer       ----------- ⑤
```

①コモンダイアログボックスから取得したファイル名を格納する変数
②セルデータを格納する変数。文字列型を指定し、さらに10文字分のサイズを指定して宣言する
③データのサイズを格納する変数。Long型で宣言する
④ランダムアクセスで書き込む際に使用するレコード番号用変数
⑤For...Nextループで使用するカウンタ変数

宣言したら、レコード番号用変数「RecNo」を初期化しておきます。

```
RecNo = 1
```

4 次に、コモンダイアログボックスを表示します。まず、［DialogTitle］プロパティに、ダイアログボックスのタイトルとなる文字を設定します。ファイルを［開く］と［保存］は同じスタイルのダイアログボックスになるので、それぞれのタイトルを変えて表示するようにします。

［保存］ダイアログボックスを表示するには、「ShowSave」メソッドを実行します。このプロパティ設定とメソッドの実行で、［ファイル名を付けて保存］ダイアログボックスが表示されます。

```
With CommonDialog1
    .DialogTitle = "データの保存"
    On Error GoTo BACK
    .ShowSave
End With
```

なお、ShowSaveメソッドの前に、On Errorステートメントを組み込んでおきます。これは、ダイアログボックスの［キャンセル］ボタンがクリックされたときに、ラベル「BACK」にジャンプし何も処理をせずにプロシージャを終了させるためです。

データの保存

保存する場所(I): (C:)

Acrobat3 / Aol30I / Kpcms / Mouse / My Documents / Program Files / Quickcmd / Realmode / Windows / Yamaha

ファイル名(N):　保存(S)

ファイルの種類(T): データファイル (*.eca)　キャンセル

読み取り専用ファイルとして開く(R)

［ファイル名を付けて保存］ダイアログボックス

ワンポイントアドバイス

通常、文字列型変数を宣言すると、文字列の長さを気にせずに格納できる、「可変文字列型変数」として使用できます。

しかし、格納する文字数を制限したい場合は、文字数を指定して変数を宣言できます。その場合は、「AS String」のあとに「* 10」のように、確保したい文字数を「*」を付けて記述します。データベースプログラムのように、フィールドのデータ数が決められている場合は、このような変数宣言の仕方をします。変数の文字数を指定した場合は、指定した文字数以上のデータを入力しても、この文字数分だけが格納されます。

次のコードは、８文字に設定した変数に９文字代入していますが、実際には８文字分しか格納されません。

```
Dim Data As String * 8
Data = "123456789"
MsgBox Data
```

Project1

12345678

OK

８文字しか表示されない

5　［ファイル名を付けて保存］ダイアログボックスが表示され、ファイル名とフォルダ名が指定されると、コモンダイアログボックスの［filename］プロパティに、このファイル名がフルパスで格納されます。これを、変数「GetFileName」に格納しておきます。

```
GetFileName = CommonDialog1.filename
```

また、確保した10文字分の文字列型変数「Data」のバイトサイズを、LenB関数で計算して、変数「DataByte」に格納します。

LenB関数は、引数に与えられた文字列の文字数を、バイト数に変換して返してくれる関数です。戻り値はLong型になりますので、受け取る変数もLong型で宣言しておきます。

```
DataByte = LenB(Data)
```

ファイル名とファイルに格納するデータのサイズが決まったら、Openステートメントを、ランダムモードで実行します。Openステートメントをランダムモードで実行する場合は、「For Random」を指定し、最後に「Len」で1レコードあたりのバイト数を指定します。ここでは、ファイル名に「GetFileName」を、「Len」に「DataByte」の変数を指定して実行します。

存在しないファイル名を指定してOpenステートメントを実行すると、新規にそのファイルが作成されます。

```
Open GetFileName For Random As #1 Len = DataByte
```

6

Openステートメントでファイルを開いたら、セルデータを1つずつ書き出していきます。ランダムアクセスでのファイルの書き出しは、Putステートメントを使用します。このステートメントは、書き出すデータとそのサイズ、そしてレコード番号を指定して実行します。

セル番地の参照は、1行目の1列から順番に行います。1列目の最後の列のデータにアクセスすると、次の行の1列目に移動します。そのため、2つのFor...Nextループで［Col］と［Row］プロパティに値を設定し、［Text］プロパティの値を変数「Data」に格納します。ループはネスト（次ページ参照）させ、中のループが列（Col）の、外のループが行（Row）の値を設定するようにします。

```
For i = 1 To 9
    For j = 1 To 5
        With MSFlexGrid1
            .Col = j
            .Row = i
            Data = .Text
        End With
```

7

セルの値を変数「Data」に格納したら、Putステートメントを使ってセルデータをファイルに書き出します。セルデータは、セル1つ1つの値を順番に書き出しますので、中のループでPutステートメントを実行します。

```
Put #1, RecNo, Data
RecNo = RecNo + 1
```

Putステートメントには、Openステートメントで使用したファイル番号を指定します。そして、レコード番号は変数「RecNo」を使用し、ループが1回まわるごとに1つ増加させて、すべてのセルのデータが自動で格納できるようにしています。

ファイルに確保される1データあたりのバイト数は、Openステートメントの「Len」で指定したバイト数「Len = DataByte」が割り当てられます。そのため、この「Len」で指定するバイト数は、実際のデータが消費するバイト数以上のバイト数を指定してください。

【Putステートメントの書式】

```
Put [#]filenumber, [recnumber], varname
```

filenumber	開いているファイルのファイル番号を指定します。Openステートメントで指定した番号と同じ番号を使います。
recnumber	書き込むデータのレコード番号を指定します。任意の番号を指定できます。ファイルの中のデータへはこの番号を使ってアクセスすることができ、ファイルからデータを読み出すときにも使用します。
varname	書き込むデータの変数を指定します。

＜例＞
次の記述は、ファイルの先頭から2番目の位置にデータを書き込みます。

```
Put #1, 2, Data
```

! ワンポイントアドバイス

ネストというのは、「入れ子」という意味です。プログラムでFor...Nextループの中に、さらにFor...Nextループを組み込むような状態を「ネスト」と呼んでいます。

```
For ~
    For ~
       ・
       ・
       ・
    Next
Next
```

ループ部分

ネスト部分

8 これで、データをファイルに書き込む操作が終わりました。後は、Closeステートメントでファイルを閉じます。また、On Errorステートメントでエラー処理を組み込んでいるので、正常終了の処理であるExit（Sub）ステートメントを記述しておきます。そして、最後に、エラー処理のためのラベル「BACK:」を設定しておきます。

［ファイル名を付けて保存］ダイアログボックスの［キャンセル］ボタンがクリックされると、ファイル名の指定がないままOpenステートメントが実行されてしまい、エラーになります。そこで、On Errorステートメントでこのラベルに処理をジャンプさせますが、［キャンセル］ボタンがクリックされて発生したエラーなので、特に何の処理もせずにプロシージャを終了させるので、ラベルの記述のみにしています。

```
Close #1
Exit Sub

BACK:
```

9 以上で、入力されたデータにファイル名を付けて保存する処理が完成です。

プログラムを動作させ、セルに値を入力して、ファイルを保存してください。このデータファイルは、基本的な部分はテキスト形式で保存されていますから、Windowsのメモ帳などで中身を見ることができます。

まとめ

ランダムアクセスモードは、シーケンシャルモードと違い、あらかじめデータに確保するバイト数がわかっているため、データを任意の位置に書き込むことができます。データベースプログラムのように、不連続なデータを格納するのに便利なモードです。必ず、書き込むデータのサイズは、Openステートメントの「Len」で指定したバイト数以内に収まるようにしてください。指定した以上のバイトは無視されます。

また、Openステートメントでファイルを開いたら、Closeステートメントでファイルを閉じてください。

練習問題

Q1　作成したプログラムでは、既にあるファイルと同じファイル名を使用して保存すると、いきなりそのファイル名で上書き保存してしまいます。そこで、すでに同じファイル名がある場合は、上書き保存していいかどうかを確認する処理を組み込んでください。

ファイルの上書き確認

すでに同じファイルが存在します。
上書き保存しますか

はい(Y)　いいえ(N)

【ヒント】

ファイルが存在するかどうかを調べるには、Dir関数を使うと便利です。この関数は、引数にファイル名をフルパスで指定して実行すると、ファイルが存在すればそのファイル名を返してきます。ファイルが存在しなければ、空の文字（""）を返してきます。この戻り値を使って、ファイルが存在するかどうかを判断するといいでしょう。

【Dir関数の書式】

```
Dir [(pathname[, attributes])]
```

pathname	パス名を指定します。省略可。
attributes	ファイル属性を値や定数で指定します。省略可。

解答は巻末に→

Lesson 4

第１日
４時限目

かんたん表計算ソフトの作成

Lesson4.xls
VB6CD2
Lesson04

ファイルからデータを読み込む

ランダムアクセスモードでのファイルの読み込みを使って、データを保存したファイルを開き、セルに値を組み込みます。ファイル名の取得は、コモンダイアログボックスの［ファイルを開く］ダイアログボックスを利用します。ランダムアクセスモードでは、レコード番号を指定して、ファイル内の任意の位置のデータを取り出すことができます。

今回作成するプログラム

簡単表計算ソフト
ファイル
Input

	A	B	C	D
1	123	223	123	
2				

［ファイルを開く］ダイアログボックスを使って、保存されているデータをセルに組み込む

ファイルを開く
ファイルの場所(I): (C:)
Acrobat3
Aol30I
Kpcms
Mouse
My Documents
Program Files
Quickcmd
Realmode
Windows
Yamaha
mydata.eca
ファイル名(N): mydata.eca
ファイルの種類(T): データファイル (*.eca)
開く(O)
キャンセル
読み取り専用ファイルとして開く(R)

POINT

ランダムアクセスモードで、ファイルからレコード番号を指定してデータを取り出します。この処理は、Get ステートメントを使用して行います。また、コモンダイアログボックスの［ファイルを開く］ダイアログボックスの使い方を覚えると、ファイル名の取得がとても簡単に行えます。

コントロールとプロパティの設定

※前回作成した「Lesson3.vbp」を開いてください。また、今回はコードを追加するだけなので、ユーザーインターフェイスの変更はありません。「Lesson3.vbp」ファイルは、付録CD-ROMの「Lesson03」フォルダに収録されています。

コードの記述

Form1:IDM_OPEN-Click

Lesson4 - Form1 (ｺｰﾄﾞ)

IDM_OPEN　Click

```
Private Sub IDM_OPEN_Click()
    Dim GetFileName As String
    Dim Data As String * 10
    Dim DataByte As Long
    Dim RecNo As Integer
    Dim i As Integer, j As Integer, k As Integer
    Dim cc As String
    Dim Data2 As String
    Dim Num As Integer

    RecNo = 1

    With CommonDialog1
        .DialogTitle = "ファイルを開く"
        On Error GoTo BACK
        .ShowOpen
    End With

    GetFileName = CommonDialog1.FileName
    DataByte = LenB(Data)

    Open GetFileName For Random As #1 Len = DataByte

    For i = 1 To 9
        For j = 1 To 5
            Get #1, RecNo, Data

            Num = Len(Data)
            For k = 1 To Num
```

```
            cc = Mid(Data, k, 1)
            If cc <> " " Then
                Data2 = Data2 & cc
            End If
        Next

        With MSFlexGrid1
            .Col = j
            .Row = i
            .Text = Data2
        End With

        Data2 = ""
        RecNo = RecNo + 1
    Next
Next

Close #1
Exit Sub

BACK:

End Sub
```

解説

1 ランダムアクセスモードでのファイルの読み出しは、前回のレッスンで作成したファイルへの書き出しとほぼ同じような処理になります。ファイルへの書き出しがPutステートメントを使ったのに対し、読み出しはGetステートメントを使用します。

最初に、変数の宣言をしますが、この宣言はファイル書き出しのコードと同じ変数を使用します。

```
Dim GetFileName As String
Dim Data As String * 10
Dim DataByte As Long
Dim RecNo As Integer
Dim i As Integer, j As Integer, k As Integer
```

また、セルに値を入力する際に行う空白除去の処理に使用する変数を宣言します。

```
Dim cc As String
Dim Data2 As String
Dim Num As Integer
```

2つのプロシージャでまったく同じ変数を使う場合は、モジュールレベルの変数（P.53参照）として宣言してもかまいません。変数宣言をしたら、レコード番号の変数「RecNo」を初期化しておきます。

```
RecNo = 1
```

2　続いて、コモンダイアログボックスコントロールを、［ファイルを開く］ダイアログボックスで表示します。メソッドは「ShowOpen」を使います。また、ダイアログボックスのタイトルも「ファイルを開く」に変更します。

［キャンセル］ボタンがクリックされたときのエラー処理（On Error GoTo BACK）を組み込むのを忘れないでください。

```
With CommonDialog1
    .DialogTitle = "ファイルを開く"
    On Error GoTo BACK
    .ShowOpen
End With
```


表示された［ファイルを開く］ダイアログボックス

3　［ファイルを開く］ダイアログボックスを表示したら、ここに入力されたファイル名を、［filename］プロパティから取得します。

```
GetFileName = CommonDialog1.filename
```

また、LenB関数を使って、1レコード分のデータのバイト数を計算し、Openメソッドを実行します。この処理は、データ書き込みの処理とまったく同じです。

```
DataByte = LenB(Data)

Open GetFileName For Random As #1 Len = DataByte
```

4 ファイルを開いたら、Getステートメントでファイルからデータを取得します。Getステートメントは、指定したレコード番号のデータをファイルから読み込むステートメントです。読み込むファイル番号とレコード番号、読み込んだデータを格納する変数名を指定します。ランダムアクセスモードとGetステートメントを使用すると、ファイル内の任意の位置にあるデータを取り出すことができます。

ここでは、まずGetステートメントでレコード番号を1から順番に読み込んで、フレキシブルグリッドコントロールのセルに順番に格納しています。書き込み処理と同じように、内側のFor...Nextループで列番号を、外側のループで行番号を指定し、左上のセルから列方向に順番にデータを格納していきます。この順番は、データ書き込みの処理と同じにしてください。そうしないと、セルのポジションと値が保存したときと一致しなくなります。

```
For i = 1 To 9
    For j = 1 To 5
        Get #1, RecNo, Data
```

【Getステートメントの書式】

```
Get [#]filenumber, [recnumber], varname
```

filenumber	開いているファイルのファイル番号を指定します。Openステートメントで指定した番号と同じ番号を使います。
recnumber	読み出すデータのレコード番号を指定します。ファイルの中のデータはこの番号を使ってアクセスすることができ、任意の番号を指定することで任意の位置のデータを読み出すことができます。
varname	読み込んだデータを格納する変数を指定します。

＜例＞

次の記述は、ファイルの先頭から2番目の位置にあるデータを読み込みます。

```
Get #1, 2, Data
```

5 このとき、たとえば3桁の数字の場合は、テキストファイルからそのまま読み込んでセルに転送すると、数字がセルの中央に表示されてしまいます。

これは、ファイルのデータをセルに転送する際に、10桁に満たない数字は、後ろに空白を付けて10文字になるように調整されて、いっしょにセルに転送されてしまうからです。

したがって、データをセルに転送する前に、数字以外の空白を取り除いて、数字だけをセルに転送するような文字列処理を行ってあげます。

まず、Getステートメントで読み込んだセル1つ分のデータが何文字あるのかをLen関数で数えます。

次に、For...NextステートメントとMid関数を組み合わせて、データの先頭から1文字ずつ取り出します。取り出した文字は変数「cc」に格納し、Ifステートメントで空白でないかどうかをチェックします。そして、空白でなければ、変数「Data2」に格納します。

この処理を、データの全文字に対し行えば、空白を取り除いた数字だけが変数「Data2」に代入されます。

```
Num = Len(Data)
For k = 1 To Num
    cc = Mid(Data, k, 1)
    If cc <> " " Then
        Data2 = Data2 & cc
    End If
Next
```

あとは、この数字をフレキシブルグリッドコントロールのTextプロパティに代入すれば、セルに右詰めで数字が表示されます。

```
With MSFlexGrid1
     .Col = j
     .Row = i
     .Text = Data2
End With
```

1データ分使用した変数「Data2」は、次のデータのために""を代入し初期化しておきます。そして、レコード番号を1つ増やし、2番目以降のデータに対し同じ処理を繰り返します。

```
        Data2 = ""
        RecNo = RecNo + 1
    Next
Next
```

6 最後に、ファイルをクローズし、Exitステートメントで正常な処理のプロシージャを終了します。また、ラベル「BACK」を設定し、エラー処理の終了処理を作成して完成です。

```
Close #1
Exit Sub

BACK:
```

7 プログラムを起動し、データを入力したのちにファイルを保存します。一度プログラムを終了し、再度同じファイルを開くと、保存したときと同じセル位置にデータが格納されます。

例として、「Lesson04」フォルダの「test.eca」を開いてみてください。

ワンポイントアドバイス

フレキシブルグリッドコントロールの「CellAlignment」プロパティを使用すると、セル内に表示する文字の表示位置を設定できます。このプロパティはデザイン時には設定できず、コードから行います。ただし、今回のように空白を含む文字列の場合は、空白も文字として扱われてしまいます。

プロパティの設定値は定数で、次の値を使用します。

定数	値	内容
flexAlignLeftTop	0	セルの内容は左側の上に配置されます。
flexAlignLeftCenter	1	文字列の既定値です。セルの内容は左側の中央に配置されます。
flexAlignLeftBottom	2	セルの内容は左側の下に配置されます。
flexAlignCenterTop	3	セルの内容は中央の上に配置されます
flexAlignCenterCenter	4	セルの内容は中央の中央に配置されます。
flexAlignCenterBottom	5	セルの内容は中央の下に配置されます。
flexAlignRightTop	6	セルの内容は右側の上に配置されます。
flexAlignRightCenter	7	数値の既定値です。セルの内容は右側の中央に配置されます。
flexAlignRightBottom	8	セルの内容は右側の下に配置されます。
flexAlignGeneral	9	セルの内容は通常の方法で配置されます。つまり、文字列は左側の中央に配置され、数値は右側の中央に配置されます。

ワンポイントアドバイス

変数を「モジュールレベル変数」として宣言すると、そのフォームモジュール内のすべてのプロシージャから宣言した変数を参照することができます。変数の値を共有したいときにはモジュールレベル変数として宣言するのがよいでしょう。

反対にプロシージャ内で宣言する変数は「ローカル変数」と呼ばれ、そのプロシージャ内だけで有効な変数になります。

簡単表計算ソフト

ファイル

開く

保存

Input

	A	B	C	D	合計
1					
2					
3					
4					
5					
6					
7					

1 [ファイル] メニューの [開く] を選ぶ

ファイルを開く

ファイルの場所(I): (C:)

Acrobat3
Aol30I
Kpcms
Mouse
My Documents
Program Files
Quickcmd
Realmode
Windows
Yamaha
mydata.eca

ファイル名(N): mydata.eca 開く(O)

ファイルの種類(T): データファイル (*.eca) キャンセル

読み取り専用ファイルとして開く(R)

2 ファイルを指定し...

3 [開く] をクリックする

簡単表計算ソフト

ファイル

Input

	A	B	C	D	合計
1	123	223	123		
2					
3					
4					
5					
6					
7					

データがセルに格納される

まとめ

ランダムアクセスモードでのデータの読み込みは、読み出すデータのレコード番号を指定して、Get ステートメントを実行します。

このプログラムでは、ファイルを先頭から順番にアクセスしてデータを読み込みましたが、レコード番号を上手に使えば、Microsoft Access のように、データを選んで入力・更新することができます。

練習問題

Q1　このプログラムでは、Open ステートメントで使用するファイル番号を、直接「#1」と指定して使用していますが、他のアプリケーションによってこの番号が使われている場合があります。

そこで、空いているファイル番号を返す「FreeFile」関数を使用して、現在使用可能なファイル番号を取得してから使用するようにプログラムを修正してください。FreeFile 関数の使用方法は簡単なので Visual Basic のオンラインヘルプを参照してみてください。

解答は巻末に→

COLUMN・・・・・

キーボードで操作する

例えば［Input］ボタンをクリックして入力するより、キーボードでEnterキーを押したとき、数値が入力できるようになるとよりソフトウェアらしくなります。

Enterキーが押されたことを処理に利用するには、KeyDownイベントを利用してみましょう。考え方としては「Text1」コントロール上でEnterキー（KeyCode = 13）が押されたときに［Input］ボタンと同じ動作をさせるようにします。以下のコードを追加してみましょう。

```
Private Sub Text1_KeyDown(KeyCode As Integer, Shift As _
    Integer)
    If KeyCode = 13 Then
        MSFlexGrid1.Text = Text1.Text
        Text1.Text = ""
    End If
End Sub
```

10日でおぼえる Visual Basic 6.0 実践教室

Let's VBSchool!!

第2日 (Lesson5 ~ Lesson7)
ワープロソフトの作成

Visual Basic 6.0 に装備されている、リッチテキストボックスコントロールは、テキストボックスコントロールを機能拡張して、文字の装飾やインデントの設定、フォントの選択などが行えるコントロールです。その機能は大変強力で、他のActiveX コントロールと組み合わせると、ちょっとしたワープロソフトなみのプログラムを自分で作成することができます。

このレッスンでは、リッチテキストボックスコントロール、ツールバーコントロール、イメージリストコントロールを組み合わせて、簡易ワープロソフトを作成します。

今日のレッスンで解説する項目

❑ ツールバーコントロールの設定
❑ イメージリストコントロールへのイメージの組み込み
❑ ツールバーコントロールとイメージリストコントロールの連動操作
❑ アイコンファイルのサイズ
❑ ツールバーコントロールのイベントプロシージャの処理
❑ リッチテキストボックスコントロールの設定
❑ RTF 形式でのファイルの保存と読み込み
❑ ツールバーコントロールとリッチテキストボックスコントロールの連携
❑ コモンダイアログボックスコントロールを使ったファイル名の取得と設定
❑ 選択した文字列のフォント操作と文字色の設定
❑ 選択した文字列の表示位置の操作
❑ コンボボックスコントロールとリッチテキストボックスコントロールの連携操作
❑ リッチテキストボックスコントロールの印刷機能

Lesson 5

第２日
１時限目

ワープロソフトの作成

Lesson5.vbp
VB6CD2
Lesson05

ツールバーコントロールの組み込み

まずは、フォームにツールバーを組み込みます。ツールバーコントロールを使うと、簡単にプログラムにツールバーを作成できます。また、イメージリストコントロールを使って、このツールバーにグラフィックを組み込み、Microsoft Wordのようなツールバーに仕上げます。

今回作成するプログラム

Form1

グラフィックを組み込んだツールバーを作成する

POINT

ツールバーコントロールは、１つのコントロールでいくつでもボタンを組み込むことができます。また、イメージリストコントロールと組み合わせることで、好きなグラフィックをツールバーのボタンに貼り付けることができます。

イメージリストコントロールは、複数のアイコンやビットマップを一元管理する便利なコントロールです。このコントロールを使えば、いくつものイメージファイルを持ち歩く必要はありません。

コントロールとプロパティの設定

※新しいプロジェクトファイル（標準EXE）を用意します。また、付録CD-ROMに収録されている「Material」フォルダ内のファイルを利用します。

1 ツールパレットをマウスの右ボタンでクリックし、メニューから［コンポーネント］を選ぶ

コンポーネント(O)...
タブの追加(A)...
ドッキング可能(K)
非表示(H)

1 ここをクリック

2 ［Microsoft Windows Common Control 6.0］をチェックし、［OK］ボタンをクリックする

コンポーネント
コントロール | デザイナ | 挿入可能なオブジェクト
Microsoft Rich Textbox Control 6.0
Microsoft SysInfo Control 6.0
Microsoft Tabbed Dialog Control 6.0
Microsoft Wallet
Microsoft Windows Common Controls 5.0 (SP2)
Microsoft Windows Common Controls 6.0
Microsoft Windows Common Controls-2 5.0 (SP2)
Microsoft Windows Common Controls-2 6.0
Microsoft Windows Common Controls-3 6.0
Microsoft Winsock Control 6.0
参照(B)...
選択された項目のみ(S)
Microsoft Windows Common Controls 6.0
場所:　c:¥WINDOWS¥SYSTEM¥MSCOMCTL.OCX
OK　キャンセル　適用(A)

1 ここをチェック

2 ここをクリック

ツールパレットにコモンコントロールが追加される

(SP3)について

Visual Basicでは「Service Pack（サービスパック）」というアップデータが用意されています。アップデートされている場合、コントロール名の右端に「(SP3)」などのサービスパック名が表示されていることがあります。

フォームを広げる

フォームを広げるには、フォームの周囲に表示されているサイズ変更ハンドル（黒い四角形）を外方向にドラッグします。

3 フォームを広げ、ツールパレットの［ImageList］を使ってイメージリストコントロールを配置する

1 斜めにドラッグして配置

[…] ボタン

[…] ボタンは、そのプロパティの詳細な指定を行うダイアログボックスが用意されていることを示しています。

4 プロパティウィンドウの［(プロパティページ)］をクリックし［…］ボタンをクリックする

1 ここをクリック

2 ここをクリック

5 ［全般］タブの［16x16］をクリックする

1 ここをクリック

6 ［イメージ］タブをクリックして［ピクチャの挿入］ボタンをクリックする

プロパティ ページ
全般　イメージ
現在のイメージ
インデックス(I):　キー(K):
タグ(T):
イメージ(M):
ピクチャの挿入(P)...　ピクチャの削除(R)　イメージ数:
OK　キャンセル　適用(A)　ヘルプ

1 ここをクリック
2 ここをクリック

7 「Material」フォルダにある「Disk_3j.ico」を選び、［開く］をクリックする

図の選択
ファイルの場所(I): material
Buttonrg.ico　Clock8.ico　Doc05.ico　Filecb09.ico
Calc.ico　Copya.ico　Doc12.ico　Filecb18.ico
Camera.ico　Disk_3g.ico　Documnt.ico　Finger.ico
center.ico　Disk_3i.ico　Drive.ico　Floppy.ico
check.ico　Disk_3j.ico　Drivea.ico　Floppy_1.ico
Clipbrd2.ico　Diskcat2.ico　File07.ico　Floppy_2.ico
ファイル名(N): Disk_3j.ico　開く(O)
ファイルの種類(T): ビットマップおよびアイコン ファイル (*.bmp, *.ico)　キャンセル

1 「Disk_3j.ico」を選ぶ
2 ここをクリック

8 選択したアイコンが［プロパティページ］ダイアログの［イメージ］ボックスに追加される

タグ(T):
イメージ(M):
ピクチャの挿入(P)...　ピクチャの削除(R)
OK　キャンセル

追加されたアイコン

［ピクチャの挿入］ボタン

［ピクチャの挿入］ボタンをクリックすると、ピクチャを選択するための［図の選択］ダイアログボックスが表示されます。

「Material」フォルダの場所

［ファイルの場所］ボックスで、「Material」フォルダをコピーした場所を選択してください。

追加されたアイコン

［イメージ］ボックスに追加されたアイコンが、フォーム上のイメージリストコントロールに追加されます。

9 [キー] ボックスに「save」と入力する

1 「save」と入力

10 手順６から９を繰り返して、次のアイコンを設定し、最後に [OK] ボタンをクリックする

ファイル名	キー
File07.ico	open
Center.ico	center
Left.ico	left
Right.ico	right
Printer12.ico	print

追加されたアイコン

11 ツールパレットの［Toolbar］を使って、フォームにツールバーコントロールを配置する

1 斜めにドラッグして配置

12 ［プロパティページ］の［全般］タブで、［イメージリスト］に［ImageList1］を指定する

1 ここをクリック

2 ここをクリック

13 ［ボタン］タブをクリックし、［ボタンの挿入］をクリックする

1 ここをクリック

2 ここをクリック

［プロパティページ］ダイアログ

［プロパティページ］ダイアログボックスを表示するには、プロパティウィンドウの［(プロパティページ)］をクリックし、右欄の［...］ボタンをクリックします。

14 [キー] に「open」、[ツールヒント] に「ファイルを開く」、[イメージ] に「open」と入力する

1 「open」と入力

2 「ファイルを開く」と入力

3 「open」と入力

15 手順13と14を繰り返して、残りの5つのボタンを組み込み、最後に [OK] をクリックする

インデックス	キー	ツールヒント	イメージ
2	save	ファイル保存	save
3	center	中央揃え	center
4	left	左寄せ	left
5	right	右寄せ	right
6	print	印刷	print

1 最後に [OK] をクリック

16 [BorderStyle] プロパティを「1-ccFixedSingle」に設定する

(オブジェクト名)	Toolbar1
(バージョン情報)	
(プロパティ ページ)	
Align	1 - vbAlignTop
AllowCustomize	True
Appearance	1 - cc3D
BorderStyle	1 - ccFixedSingle
ButtonHeight	0 - ccNone
ButtonWidth	1 - ccFixedSingle
DragIcon	(なし)
DragMode	0 - vbManual
Enabled	True
Height	420
HelpContextID	0

1 ここをクリック

2 ここをクリック

17 完成したツールバー

Form1

右寄せ

解説

1 イメージリストコントロールは、複数のアイコンやビットマップイメージを、1つのコントロールに格納して管理するコントロールです。使い方は、画面の操作で説明したとおり、実に簡単です。初期設定は、コントロールの[(プロパティページ)]で行います。

①イメージリストコントロールをクリックして、プロパティウィンドウから[プロパティページ]ダイアログボックスを開きます。

②最初に、格納するイメージのサイズを指定します。「16×16」というのがそのサイズで、これは「縦横が16ドットのサイズ」という意味になります。通常、アプリケーションのアイコンは、「32×32」ドットのサイズで、タスクバーやフォームに表示されるアイコンのサイズが「16×16」ドットに相当します。
ここでは、小さいサイズのツールバーを作成するために、「16×16」ドットのサイズを指定しました。サイズを設定すると、このイメージリストコントロールに追加するイメージは、すべてこのサイズに統一されます。

プロパティ ページ
全般 | イメージ | 色
16 x 16(1)
32 x 32(3)
48 x 48(4)
ユーザー設定(C)
MaskColor の使用(U)
高さ(H): 16
幅(W): 16
OK | キャンセル | 適用(A) | ヘルプ

③次に[イメージ]タブの[ピクチャの挿入]ボタンで、アイコンまたはビットマップイメージを挿入します。このときに、「インデックス」の数字が自動的に増加されていきます。「インデックス」は、あとでこのイメージを取り出すときに参照するインデックス番号として使われます。

④ピクチャを挿入したら、「キー」に名前を付けます。イメージリストコントロールに格納したイメージを使用する場合は、インデックス番号を使うか、またはこの「キー」に記述した名前で参照します。イメージに合った「キー」の名前を付けたほうが、あとで参照するときに番号よりもわかりやすくなります。

この操作で、イメージリストコントロールに複数のイメージを格納して、他のコントロールで自由にこれらのイメージを使用できます。

ワンポイントアドバイス

イメージリストコントロールに追加するイメージが「32 × 32」ドットのサイズなのに、イメージリストコントロールで指定するサイズが「16 × 16」ドットの場合は、イメージは自動的に縮小されて格納されます。

2

ツールバーコントロールは、プログラムにツールバーを組み込んでくれるコントロールです。これまでは、自分でコマンドボタンを組み合わせて作成していましたが、ツールバーコントロールができてからは、このコントロールのプロパティを設定するだけで、簡単に作成できるようになりました。また、イメージリストコントロールと組み合わせると、グラフィック付きのツールバーが作成できます。

このコントロールも、プロパティページで初期設定を行います。

①［全般］タブにある［イメージリスト］で、ボタンに組み込むイメージリストコントロールをリストボックスから選択します。

②［ボタン］タブをクリックし、［ボタンの挿入］ボタンをクリックして、ツールバーコントロールにボタンを追加します（この操作を行わないと、ツールバーコントロールにボタンが組み込めません）。
ボタンが追加されると、「インデックス」の数字が増加します。この数字で、ユーザーが選択したボタンの種類を判別することができます。

③「キー」は、ボタン1つ1つに固有の名前を設定します。
「キー」に任意の文字を設定すると、ユーザーが選択したボタンの種類をこの文字でも把握できます。イメージリストコントロールと同様、数字ではなく文字でボタンを識別できるので、ボタンの種類がわかりやすくなります。

［キー］に文字を設定すると、ボタンの種類がわかりやすくなる

④「イメージ」に、イメージリストコントロールに組み込んだイメージのインデックス番号またはキーを記述すると、そのイメージをボタンに組み込めます。
また、「キャプション」に文字を記述すれば、ボタンの表面にイメージと文字を同時に表示できます。
グラフィック表示のボタンの場合は、イメージリストコントロールのイメージサイズによって、ボタンのサイズが変わります。

Open 「キャプション」だけを設定

「イメージ」だけを設定

Open 両方を設定

⑤［全般］タブの［ヒントを表示］にチェックを付け、［ボタン］タブの［ツールヒント］に簡単な説明を入力すれば、ツールバーコントロールのボタンにツールヒントを設定できます。

このような手順を踏んで、プログラムにツールバーコントロールを組み込みます。

「ツールヒント」を表示するツールバー

ワンポイントアドバイス

フォームに配置したツールバーコントロールは、[Align] プロパティを設定すると、自動的にその位置に配置されます。[Align] プロパティの値を [0 – vbAlignNone] に設定すると、フォームの好きな位置にツールバーコントロールを配置できます。

3　ツールバーコントロールのいずれかのボタンが、ユーザーによってクリックされると、コントロールには「ButtonClick」というイベントが発生します。このイベントに対応しているイベントプロシージャは、呼び出されると、引数に「Button」オブジェクトの参照が格納されます。

そして、このオブジェクトの［Key］プロパティを参照すると、どのボタンがクリックされたのかがわかるようになっています。［Key］プロパティは、ツールバーコントロールのプロパティページで設定した「キー」の値を格納しているプロパティです。

では、簡単なコードを作成して、ツールバーコントロールのクリックされたボタンの種類を把握してみましょう。まず、フォームに配置したツールバーコントロールをダブルクリックして、ButtonClickイベントプロシージャを表示します。そして、次のコードを入力します。

```
Private Sub Toolbar1_ButtonClick(ByVal Button As _
    MSComctlLib.Button)
    MsgBox Button.Key
End Sub
```

プログラムを実行してみます。ツールバーコントロールのいずれかのボタンを1つクリックすると、そのボタンに設定した「キー」が表示されます。このように、Buttonオブジェクトの［Key］プロパティを使うことで、どのボタンがクリックされたのかを判断できます。

ワンポイントアドバイス

今回使用した「Microsoft Windows Common Control6.0」は、Visual Basic5.0に搭載されていた「Microsoft Windows Common Control5.0」から、イベントプロシージャの引数が変更されているものがあります。

したがって、「Microsoft Windows Common Control5.0」を使って作成したコードは、そのままでは「Microsoft Windows Common Control6.0」では使用できませんので注意してください。

■「Microsoft Windows Common Control5.0」のツールバーコントロールのButtonClickイベントプロシージャ

```
Private Sub Toolbar1_ButtonClick(ByVal Button As _
                                  ComctlLib.Button)
```

■「Microsoft Windows Common Control6.0」のツールバーコントロールのButtonClickイベントプロシージャ

```
Private Sub Toolbar1_ButtonClick(ByVal Button As _
                                  MSComctlLib.Button)
```

※引数Buttonのデータ型名（オブジェクト名）が違っている

まとめ

イメージリストコントロールは、他のコントロールに複数のイメージを提供するコントロールで、イメージファイルの一元管理ができる優れた機能を持っています。
また、ツールバーコントロールは、プログラムに簡単にグラフィック付きのツールバーを装備できる便利なコントロールです。
いずれも、「プロパティページ」という新しいプロパティ設定機能を持ったコントロールです。このプロパティページは、プロパティウィンドウに比べプロパティの設定がわかりやすくなっていますが、プロパティ名との関連がわかりづらい面を持っています。

練習問題

Q1　新しいプロジェクトファイルを用意し、フォームにイメージリストコントロールとツールバーコントロールを組み込んでください。

①使用するイメージは、付録ディスクの「material」フォルダにある「244.ico」から「247.ico」の４つのアイコンを使用します。
②イメージリストコントロールのイメージサイズを「32x32」で組み込んでください。

解答は巻末に→

COLUMN

ツールバーのカスタマイズ

ツールバーコントロールは、デフォルトで、ツールバー項目をカスタマイズする機能を持っています。プログラム実行後、ツールバーの空いている（ボタンの無い）部分をダブルクリックしてください。ボタンをカスタマイズするためのダイアログボックスが表示されます。

ツールバーをダブルクリックすると項目をカスタマイズ可能

Lesson 6

第２日
２時限目

ワープロソフトの作成

Lesson6.vbp
VB6CD2
Lesson06

リッチテキストボックスコントロールの配置

このレッスンでは、フォームにリッチテキストボックスコントロールとコモンダイアログボックスコントロールを配置し、それぞれのプロパティを設定します。また、リッチテキストボックスコントロールに入力した文字列をファイルに保存し、またファイルから読み込む機能を、それぞれのプロシージャとして作成し、ツールバーコントロールのButtonClickイベントに組み込みます。

今回作成するプログラム

リッチテキストボックスに入力された文字列をファイルとして保存し、保存されたファイルを開く機能をボタンに組み込む

POINT

リッチテキストボックスコントロールは、コントロールに入力された文章（文字列）を、RTF形式とテキスト形式の両方のフォーマットで保存できます。純粋な文字データだけのテキスト形式に対し、RTF形式は、フォント属性や段落、インデントなどの文書整形の属性をいっしょに保存できるファイル形式です。

読み書きするファイルの名前の設定・取得には、コモンダイアログボックスコントロールを使用して行います。

コントロールとプロパティの設定

※Lesson5で作成したプログラム（Lesson5.vbp）を開いてください。

1 ツールパレットをマウスの右ボタンでクリックし、メニューから［コンポーネント］を選ぶ

2 ［Microsoft Common Dialog Control 6.0］をチェックする

3 ［Microsoft Rich Textbox Control 6.0］をチェックし、［OK］ボタンをクリックする

Lesson5.vbpファイル

Lesson5.vbpファイルは、付録CD-ROMの「Lesson05」フォルダにも収録されています。

コンポーネントが追加される

手順1～3により、［CommonDialog］ボタンと［RichTextBox］ボタンがツールパレットに追加されます。

4 ツールパレットの［CommonDialog］を使って、コモンダイアログボックスコントロールを配置する

1 斜めにドラッグして配置

Form1
ファイル

5 ［プロパティページ］の［開く/保存］タブで、［フィルタ］を［リッチテキスト(*.rtf)¦*.rtf］に設定する

［プロパティページ］ダイアログ

［プロパティページ］ダイアログボックスを表示するには、プロパティウィンドウの［(プロパティページ)］をクリックし、右欄の［...］ボタンをクリックします。

プロパティページ
開く/保存 | 色 | フォント | 印刷 | ヘルプ
タイトル(T):
ファイル名(I):
初期ディレクトリ(D): c:¥
フィルタ(R): リッチテキスト(*.rtf)|*.rtf
フラグ(F): 0
既定の拡張子(U):
最大ファイル サイズ(M): 260
フィルタ インデックス(N): 0
キャンセル時にエラー(E)
OK キャンセル 適用(A) ヘルプ

1 ここを設定

6 ［既定の拡張子］に「rtf」を設定し、［OK］ボタンをクリックする

プロパティページ
開く/保存 | 色 | フォント | 印刷 | ヘルプ
タイトル(T):
ファイル名(I):
初期ディレクトリ(D): c:¥
フィルタ(R): リッチテキスト(*.rtf)|*.rtf
フラグ(F): 0
既定の拡張子(U): rtf
最大ファイル サイズ(M): 260
フィルタ インデックス(N): 0
キャンセル時にエラー(E)
OK キャンセル 適用(A) ヘルプ

1 「rtf」を設定

2 ここをクリック

7 ツールパレットの［RichTextBox］を使って、リッチテキストボックスコントロールを配置する

Form1
RichTextBox1

1 斜めにドラッグして配置

8 ［プロパティページ］の［全般］タブで、次の項目がチェックされていることを確認する

［ポップアップメニューの表示］、［イベント認識］、［複数行の編集］、［選択の自動解除］

9 ［描画スタイル］タブで、［スクロールバー］を［3 − rtfBoth］に設定し、［OK］をクリックする

1 ここをクリック
2 ここをクリック
3 ここをクリック

文字色の一覧を表示する

ボタンの右側にある下向き三角をクリックすると、色の一覧が表示されます。

［プロパティページ］ダイアログ

［プロパティページ］ダイアログボックスを表示するには、プロパティウィンドウの［(プロパティページ)］をクリックし、右欄の［...］ボタンをクリックします。

10 [Text] プロパティの文字列を削除する

1 文字列を削除

11 メニューエディタを起動し、メニューを作成する

キャプション	名前
ファイル	IDM_FILE
新規作成	IDM_NEW
開く	IDM_OPEN
上書き保存	IDM_SAVE
-	separator
印刷	IDM_PRINT
-	separator2
終了	IDM_EXIT

設定したメニューエディタ

メニューエディタを起動する

メニューエディタを起動するには、[標準] ツールバーの [メニューエディタ] ボタンをクリックします。

メニューを作成する方法

メニューを作成する詳しい方法については、「Lesson2 合計計算処理の作成」を参照してください。

コードの記述

Form1:Toolbar1-ButtonClick

Lesson6 - Form1 (ｺｰﾄﾞ)

Toolbar1 | ButtonClick

```
Private Sub Toolbar1_ButtonClick(ByVal Button As _
    MSComctlLib.Button)
    Select Case Button.Key
        Case "save"
            Call FileSave

        Case "open"
            Call FileLoad
    End Select
End Sub
```

Form1:(General)-FileSave

Lesson6 - Form1 (ｺｰﾄﾞ)

(General) | FileSave

```
Public Sub FileSave()
    Dim Fname As String

    With CommonDialog1
        .DialogTitle = "ファイルの上書き保存"
        .ShowSave
        Fname = .filename
    End With
    RichTextBox1.SaveFile Fname, rtfRTF
End Sub
```

Form1:(General)-FileLoad

Lesson6 - Form1 (ｺｰﾄﾞ)

(General) | FileLoad

```
Public Sub FileLoad()
    Dim Fname As String

    With CommonDialog1
        .DialogTitle = "ファイルを開く"
        .ShowOpen
        Fname = .filename
    End With
    RichTextBox1.LoadFile Fname, rtfRTF
End Sub
```

Form1:IDM_EXIT-Click

Lesson6 - Form1 (コード)

IDM_EXIT | Click

```
Private Sub IDM_EXIT_Click()
    Unload Form1
    End
End Sub
```

Form1:IDM_NEW-Click

Lesson6 - Form1 (コード)

IDM_NEW | Click

```
Private Sub IDM_NEW_Click()
    RichTextBox1.TextRTF = ""
End Sub
```

Form1:IDM_OPEN-Click

Lesson6 - Form1 (コード)

IDM_OPEN | Click

```
Private Sub IDM_OPEN_Click()
    Call FileLoad
End Sub
```

Form1:IDM_SAVE-Click

Lesson6 - Form1 (コード)

IDM_SAVE | Click

```
Private Sub IDM_SAVE_Click()
    Call FileSave
End Sub
```

解 説

1　コードを解説する前に、リッチテキストボックスコントロールの機能の概要を説明しておきます。

リッチテキストボックスコントロールは、これまでVisual Basicに用意されていたテキストボックスコントロールを更に機能拡張したコントロールです。テキストボックスコントロールが、純粋なテキスト文書の入力・編集を行うのに対し、リッチテキストボックスコントロールでは、フォントや文字色、段落やインデントなどの文字装飾・レイアウト編集機能が付加され、テキストボックスコントロールよりも高度な編集操作ができるようになりました。また、リッチテキストボックスコントロールでは、これらの編集属性までもファイルに保存できるようになりました。

コントロールに入力された文章の保存・読み出し機能に、テキスト形式以外に「RTF」（リッチテキスト）フォーマットが採用されています。このフォーマットは、Windowsに付属するワードパッドやMicrosoft Wordでも採用されている文書フォーマットです。したがって、このコントロールを使用すれば、Wordと同じ基本性能を持った機能を、自分のプログラムに組み込むことができます。

2　リッチテキストボックスコントロールには、高度な編集機能がプロパティやメソッドを通じて実装されています。コントロールに入力されたテキストは、自由に選択することができ、選択範囲のコピー・ペースト機能ややり直し機能（アンドゥ）もあらかじめ組み込まれています。

プログラム作成操作で紹介した、プロパティページにある［ポップアップメニューの表示］チェックボックスをオンにしておくと、これらの機能をマウスの右ボタンで表示されるポップアップメニューから利用することができます。もちろん、これらのコピー・ペースト機能は、他のソフトとの間でも有効で、面倒な設定をすることなく、コントロールを組み込むだけで実現できます。

また、リッチテキストボックスコントロール内に、イメージファイルを編集文書に組み込むことができ、グラフィックソフトとの間でコピー・貼り付けが可能になっています。

ポップアップメニューが標準装備

イメージも組み込め、専用のメニューに切り替わる

3 リッチテキストボックスコントロールの、[プロパティページ] にある [描画スタイル] タブの [スクロールバー] を設定しておくと、コントロール内の文章がコントロールの枠を超えると、自動的にスクロールバーが表示されます。

スクロールバーが自動的に表示される

プロパティページにある [複数行の編集] にチェックを付けると、[MultiLine] プロパティに「True」が設定され、複数行のテキスト編集が可能になります。

また、リッチテキストボックスコントロールが扱える最大文字数のバイト数は、[MaxLength] プロパティで設定できるようになっています。そして、この値が「0」であれば、使用するコンピュータのメモリが許す限りのバイト数が使用できます。

4 このプログラムでは、コモンダイアログボックスコントロールを使用して、リッチテキストボックスコントロールで編集したファイルの読み書きを行っています。コモンダイアログボックスコントロールは、プログラムデザイン時にプロパティページでいくつかの初期設定を行っています。

まず、ダイアログボックスに表示するファイルの種類を、「フィルタ」([Filter] プロパティ) に設定しています。今回は、RTF形式のファイルで保存するように設定したため、[Filter] プロパティに「リッチテキスト(*.rtf)|*.rtf」を設定しています (リッチテキストフォーマットのファイルの拡張子は「rtf」になります)。

また、ファイル保存時に拡張子の自動入力を行わせるため、[既定の拡張子] ([DefaultExt] プロパティ) に「rtf」の文字を設定しています。

「rtf」の文字を設定

リッチテキストを設定

5　では、コードを説明します。このレッスンでは、ファイルの保存と読み出しの処理、および新規作成の処理を作成します。ファイルの保存および読み出しは、それぞれ独立したユーザー定義のSubプロシージャとして作成します。

まず、リッチテキストボックスコントロールに入力された文字列やイメージをファイルに保存するプロシージャを作成しましょう。このプロシージャは、「FileSave」という名前で作成します。フォームのコードウィンドウを開き、最下行にこのプロシージャを作成します。このプロジェクトは単一のモジュールしか組み込みませんが、Subプロシージャにキーワード「public」を付けて作成します。

```
Public Sub FileSave()

End Sub
```

6　リッチテキストボックスコントロールは、「SaveFile」メソッドを実行するだけで、コントロールに入力された文字列やイメージをそのままファイルに保存できます。特別な処理は必要ありません。また、保存するファイル名の設定も、コモンダイアログボックスコントロールの［ファイル名を付けて保存］ダイアログボックスを使用しますので、とても簡潔なコードでこれらの処理を実行できます。

まず最初に、ファイル名を格納する変数「Fname」を文字列型で宣言します。

```
Dim Fname As String
```

次に、コモンダイアログボックスのタイトルを設定して、［ファイル名を付けて保存］ダイアログボックスを表示します。これには、コモンダイアログボックスコントロールの「ShowSave」メソッドを使用します。

表示された［ファイル名を付けて保存］ダイアログボックスにファイル名が入力されて［保存］ボタンがクリックされると、コモンダイアログボックスコントロールの［filename］プロパティに、ファイル名がフルパスで格納されますので、これを変数「Fname」に格納します。

```
With CommonDialog1
    .DialogTitle = "ファイルの上書き保存"
    .ShowSave
    Fname = .filename
End With
```

ファイルの上書き保存
保存する場所(I): Lesson06
lesson6.rtf
ファイル名(N): lesson6.rtf
ファイルの種類(T): リッチテキスト(*.rtf)
保存(S)
キャンセル
読み取り専用ファイルとして開く(R)

ファイル名がフルパスで [filename] プロパティに格納される

ワンポイントアドバイス

このダイアログボックスには、フォームのデザイン時にプロパティで設定した「ファイルの種類」(フィルタ)が実行されていますので、拡張子が「rtf」のファイルしか表示されていません。

7 保存するファイル名が確定したら、リッチテキストボックスコントロールの「SaveFile」メソッドを実行します。このメソッドは、引数にファイル名と保存するファイルの形式を指定します。

ここでは、コモンダイアログボックスコントロールから受け取ったファイル名を格納している変数「Fname」と、ファイルをRTF形式で保存するので定数「rtfRTF」を指定しています。

```
RichTextBox1.SaveFile Fname, rtfRTF
```

これで、ファイルに保存するプロシージャは完成です。第1日で使用した、ランダムファイルアクセスのような、OpenメソッドとPut・Getステートメントを使う処理は記述する必要がありません。リッチテキストボックスコントロールのSaveFileメソッドがすべてこれらの処理を行ってくれます。

【SaveFileメソッドの書式】

```
object.SaveFile pathname, filetype
```

object	必ず指定します。リッチテキストボックスコントロールのオブジェクト名を指定します。
pathname	必ず指定します。コントロールの内容を保存するファイルのパスおよびファイル名を文字列式で指定します。
filetype	省略可能です。保存するファイルのタイプを整数または定数で指定します。

定数	値	内容
rtfRTF	0	(既定値) リッチテキスト形式 (RTF)。リッチテキストボックスコントロールの内容を「.RTF」ファイルとして保存します。文字属性や段落などの編集属性、組み込まれたイメージファイルなどをすべて保存します。
rtfText	1	リッチテキストボックスコントロールの内容を、純粋なテキストファイルとして保存します。文字列だけしか保存されません。

8　今度は、ファイルから読み込むプロシージャ「FileLoad」を作成します。保存と同じように、Subプロシージャにキーワード「public」を付けて作成します。

```
Public Sub FileLoad()

End Sub
```

最初に、ファイル名を保存する変数「Fname」を宣言します。

```
Dim Fname As String
```

そして、コモンダイアログボックスコントロールの［ファイルを開く］ダイアログボックスを表示します。これには、「ShowOpen」メソッドを使用します。［ファイルを開く］ダイアログボックスが表示され、ファイル名が選択されると、コモンダイアログボックスコントロールの［filename］プロパティに、ファイル名がフルパスで格納されますので、これを変数「Fname」に格納しておきます。

```
With CommonDialog1
    .DialogTitle = "ファイルを開く"
    .ShowOpen
    Fname = .filename
End With
```

9　ファイル名が取得できたら、リッチテキストボックスコントロールの「LoadFile」メソッドを実行します。このメソッドは、前項の「SaveFile」メソッドと同様、読み込むファイル名とファイルのタイプを引数で受け取ります。

ここでは、ファイル名が格納されている変数「Fname」と、RTF形式で読み込むために定数「rtfRTF」を指定しています。

```
RichTextBox1.LoadFile Fname, rtfRTF
```

これだけの処理で、保存したファイルをリッチテキストボックスコントロールに読み込むことができます。読み込んだ内容は、自由に編集して、再度保存することができます。

ワンポイントアドバイス

RTF形式で保存したファイルを、誤ってテキスト形式で読み込むと、バイナリコードがテキストに変換されて表示されてしまいますので、注意してください。

RTF形式で保存した文書ファイルをテキストで読み込むと…

このような結果になる

10 作成した保存と読み出しのプロシージャを、ツールバーコントロールの［開く］と［上書き保存］の2つのボタンに組み込みます。

前のレッスンで説明したように、ツールバーコントロールのボタンがクリックされると、このコントロールにはButtonClickイベントが発生します。どのボタンがクリックされたのかは、イベントプロシージャの引数「Button」に格納されるButtonオブジェクトの［Key］プロパティを調べればわかります。そこで、このプロパティの値を使って、Select Caseステートメントで処理を振り分けます。

［Key］プロパティの値が「save」であれば、ファイルに保存するプロシージャ「FileSave」を、「open」であればファイルを読み出すプロシージャ「FileLoad」を、Callステートメントを使って呼び出します。

これで、ツールバーのボタンをクリックするだけで、それぞれの処理が実行されます。

```
Private Sub Toolbar1_ButtonClick(ByVal Button As _
                                  MSComctlLib.Button)
    Select Case Button.Key
        Case "save"
            Call FileSave

        Case "open"
            Call FileLoad
    End Select
End Sub
```

11

同じように、プログラムに組み込んだ［ファイル］メニューの［開く］と［上書き保存］にも、この処理を組み込みます。

［開く］メニューは「IDM_OPEN_Click」、［上書き保存］メニューは「IDM_SAVE_Click」のイベントプロシージャで、Callステートメントを使って呼び出します。

これで、ツールバーでもメニューでも、同じ処理を実行できます。

```
Private Sub IDM_OPEN_Click()
    Call FileLoad
End Sub

Private Sub IDM_SAVE_Click()
    Call FileSave
End Sub
```

12

［新規作成］メニューでは、現在リッチテキストボックスコントロールに入力されている内容を消去して、新しく入力できる状態にするための処理を作成します。この処理は至って簡単で、このメニューのClickイベントプロシージャに、リッチテキストボックスコントロールの［TextRTF］プロパティの値を ""（空白）にするコードを作成するだけです。

［TextRTF］プロパティには、テキストとともに各属性を表すコードもいっしょに格納されていますから、これをすべて空白にしてしまえば、それまで編集していた内容が破棄され、新規作成と同じ状態になります。

```
Private Sub IDM_NEW_Click()
    RichTextBox1.TextRTF = ""
End Sub
```

13

最後に、［終了］メニューに、すっかりおなじみになったプログラムの終了処理を記述して完成です。

```
Private Sub IDM_EXIT_Click()
    Unload Form1
    End
End Sub
```

実際にプログラムを動作させてください。文字の入力やイメージの貼り付け、文字列を選択してポップアップメニューでコピー・貼り付けをする、という編集操作が可能になっています。

また、編集した内容をファイルに格納したり、ファイルから読み込むことも可能になっています。

ワンポイントアドバイス

リッチテキストボックスコントロールでは、入力・編集した文章が、改行コードが入っていなくても自動的に行が折り返して表示されます。また、Ctrl+A キーで「すべてを選択」、Shift+ 矢印キーで「文字を１文字ずつ選択」などのおなじみのショートカットキーもサポートされています。アクセサリの「ワードパッド」のような機能をプログラムに付与することができます。

標準のテキストボックスコントロールで使用されているようなプロパティはほとんど、リッチテキストボックスコントロールにもサポートされており、テキストボックスコントロールを使用しているアプリケーションなら、簡単にリッチテキストボックスコントロールに移行させることができます。

まとめ

リッチテキストボックスコントロールは、あらかじめ文書の編集に必要な機能が組み込まれているコントロールです。

また、文書の保存や読み出しが簡単にできるようになっていますので、すぐに多機能の編集環境をプログラムに組み込むことができます。このコントロールを使いこなすには、どのような機能や性能を持ったプロパティとメソッドがあるのかを把握するのがポイントでしょう。

Sub ステートメントの中で、「Public」と「Private」という２種類のキーワードが出てきました。入門教室でもやりましたが、いったん、おさらいしておきましょう。

Public Sub プロシージャを宣言すると、そのプロシージャはすべてのモジュールのすべてのプロシージャから参照することができます。Private Sub プロシージャはそのプロシージャを記述したモジュール内の他のプロシージャからだけ参照することができます。

練習問題

Q1

フォーム起動時に、リッチテキストボックスコントロールが新規文書であることがわかるように、フォームのキャプションに表示するようにしてください。

また、編集した文書を保存した際に、そのファイル名とパス名をフォームのキャプションに表示するようにしてください。

また、ファイルを読み込んだ場合にも、同様の処理を加えてください。

新規文書
ファイル

新規文書の表示

C:¥WINDOWS¥ﾃﾞｽｸﾄｯﾌﾟ¥VB6実践¥付録¥Lesson06¥lesson6.rtf
ファイル

リッチテキストボックスコントロールは、コントロールに入力された文章の
字列)を、RTF形式とテキスト形式の両方のフォーマットで保存できます
RTF形式は、純粋な文字データだけのテキスト形式に、フォント属性や
落、インデントなどの文書整形の属性をいっしょに保存できるファイル形

保存した文書の
ファイル名が表示

解答は巻末に→

Lesson 7

第2日
3時限目

ワープロソフトの作成

Lesson7.vbp
VB6CD2
Lesson07

文字属性の操作

文字の入力・編集とファイルの読み書きができるようになったので、今度は、文字を選択してフォント属性の変更や文字位置の編集などができるように、機能を追加していきます。また、編集した文書を印刷する機能も組み込みます。

今回作成するプログラム

選択した文字列のフォントや配置を変更する

POINT

コンボボックスコントロールを使って、リッチテキストボックスコントロール内の選択した文字の、フォントとサイズを変更できる機能を組み込みます。

また、フレームコントロールとコモンダイアログボックスコントロールを組み合わせて、文字の色を変更できるようにします。

コントロールとプロパティの設定

※ Lesson6 の Q&A で作成した「Q&A-6.vbp」を開いてください。

Q&A-6.vbp ファイル

「Q&A-6.vbp」ファイルは、付録CD-ROMの「Q&A」フォルダに収録されています。

1 ツールパレットの［ComboBox］を使って、コンボボックスをツールバーコントロールに配置する

Form1
ファイル
Combo1

1 斜めにドラッグして配置

2 ［Sorted］プロパティを［True］に設定する

ﾌﾟﾛﾊﾟﾃｨ - Combo1
Combo1 ComboBox
全体 | 項目別

OLEDragMode	0 - 手動
OLEDropMode	0 - なし
RightToLeft	False
Sorted	True
Style	True
TabIndex	False
TabStop	True
Tag	
Text	Combo1
ToolTipText	
Top	60

1 ここをクリック
2 ここをクリック
3 ここをクリック

3 ツールバーコントロールに、コンボボックスコントロールをもう 1 つ配置する

Form1
ファイル
Combo1
Combo2

1 斜めにドラッグして配置

4 ツールパレットの［Frame］ を使って、フレームコントロールをツールバーコントロールに配置する

Form1
ファイル
Combo1
Combo2
Frame1

1 斜めにドラッグして配置

5 ［Caption］プロパティの文字を削除する

プロパティ - Frame1
Frame1 Frame
全体 | 項目別

(オブジェクト名)	Frame1
Appearance	1 - 3D
BackColor	&H00000000&
BorderStyle	0 - なし
Caption	
ClipControls	True
DragIcon	(なし)
DragMode	0 - 手動
Enabled	True
Font	MS Pゴシック
ForeColor	&H80000012&

1 文字を削除

6 ［BorderStyle］プロパティを［0 －なし］に設定する

プロパティ - Frame1
Frame1 Frame
全体 | 項目別

(オブジェクト名)	Frame1
Appearance	1 - 3D
BackColor	&H00C0C0C0&
BorderStyle	0 - なし
Caption	0 - なし
ClipControls	1 - 実線
DragIcon	(なし)
DragMode	0 - 手動
Enabled	True
Font	MS Pゴシック

1 ここをクリック
2 ここをクリック
3 ここをクリック

7 [BackColor] プロパティの [パレット] タブで、[黒] を設定する

1 ここをクリック
2 ここをクリック
3 ここをクリック
4 ここをクリック

色を選択するパレットを表示する

ボタンの右側にある下向き三角をクリックすると、色の一覧が表示されます。

8 リッチテキストボックスの [Font] プロパティで、フォントを [Arial]、サイズを [12] に設定する

1 [フォント名] と [サイズ] を設定
2 [OK] をクリック

フォントの設定

リッチテキストボックスコントロールをクリックして選択し、プロパティウィンドウの [Font]」プロパティの右欄にある [...] ボタンをクリックして [フォント] ダイアログボックスを表示します。

9 [HideSelection] プロパティを [False] に設定する

1 ここをクリック

2 ここをクリック

3 ここをクリック

10 完成したフォーム

コードの記述

Form1:Combo1-Click

Lesson7 - Form1 (コード)

Combo1 | Click

```
Private Sub Combo1_Click()
    With RichTextBox1
        .SelFontName = Combo1.Text
        .SetFocus
    End With
End Sub
```

Form1:Combo2-Click

Lesson7 - Form1 (コード)

Combo2 | Click

```
Private Sub Combo2_Click()
    With RichTextBox1
        .SelFontSize = Combo2.Text
        .SetFocus
    End With
End Sub
```

Form1:Combo2-KeyDown

Lesson7 - Form1 (コード)

Combo2 | KeyDown

```
Private Sub Combo2_KeyDown(KeyCode As Integer, Shift As Integer)
    If KeyCode = vbKeyReturn Then
        RichTextBox1.SelFontSize = Combo2.Text
    End If
End Sub
```

Form1:Form-Load

Lesson7 - Form1 (コード)

Form | Load

```
Private Sub Form_Load()
    Dim FontNum As Integer
    Dim i As Integer
    Dim FntName As String

    Form1.Caption = "新規文書"
```

```
    FontNum = Screen.FontCount
    For i = 0 To FontNum - 1
        FntName = Screen.Fonts(i)
        Combo1.AddItem FntName
        If FntName = RichTextBox1.Font Then
        Combo1.Text = FntName
    End If
    Next

    For j = 2 To 30 Step 2
        Combo2.AddItem j
        If j = RichTextBox1.Font.Size Then
            Combo2.Text = j
        End If
    Next
End Sub
```

▤Form1:Frame1-Click

Lesson7 - Form1 (ｺｰﾄﾞ)
Frame1 | Click

```
Private Sub Frame1_Click()
    With CommonDialog1
        .ShowColor
        Frame1.BackColor = .Color
        RichTextBox1.SelColor = .Color
    End With
End Sub
```

▤Form1:Toolbar1-ButtonClick

Lesson7 - Form1 (ｺｰﾄﾞ)
Toolbar1 | ButtonClick

```
Private Sub Toolbar1_ButtonClick _
                    (ByVal Button As MSComctlLib.Button)

    Select Case Button.Key
        Case "save"
            Call FileSave

        Case "open"
            Call FileLoad

        Case "center"
            RichTextBox1.SelAlignment = rtfCenter
```

```
        Case "left"
            RichTextBox1.SelAlignment = rtfLeft

        Case "right"
            RichTextBox1.SelAlignment = rtfRight

        Case "print"
            Call Printout
    End Select
End Sub
```

Form1:(General)-Printout

Lesson7 - Form1 (ｺｰﾄﾞ)

(General) | Printout

```
Private Sub Printout()
    Dim Ret As Integer

    Ret = MsgBox(Form1.Caption & " を印刷しますか？", vbYesNo, _
          "印刷の実行")
    If Ret = vbYes Then
        Form1.RichTextBox1.SelPrint Printer.hDC
    End If
End Sub
```

Form1:IDM_PRINT-Click

Lesson7 - Form1 (ｺｰﾄﾞ)

IDM_PRINT | Click

```
Private Sub IDM_PRINT_Click()
    Call Printout
End Sub
```

解 説

1 このレッスンでは、リッチテキストボックスコントロールに入力された文字を選択状態にしたときに、フォントの変更や文字色の設定、中央揃えなどの文字位置の設定操作を行う機能、そして文書をプリンタに出力して印刷する機能をプログラムに組み込みます。この機能の流れを簡単にまとめます。

①2つのコンボボックスコントロールに、コンピュータで利用可能なフォント名とサイズを設定します。利用可能なフォント名は、Windowsから自動的に取得します。

②ユーザーがリッチテキストボックスコントロールに入力した文字を選択し、コンボボックスコントロールでフォントとサイズを選びます。

③選択した文字が、コンボボックスコントロールで選んだフォントとサイズに変更されます。

フォントとサイズが変更された

④色で塗りつぶしたフレームコントロールをクリックします。

ここをクリック

⑤コモンダイアログボックスコントロールの［色の設定］ダイアログボックスが表示されるので、好きな色を選択します。

⑥文字色が指定した色に変更され、ツールバーのフレームコントロールもその色に塗りつぶされます。

文字とフレームの色が変更される

⑦再度文字を選択して、ツールバーの［中央揃え］ボタンをクリックします。その文字が画面の中央に配置されます。

ワンポイントアドバイス

このプログラムでは、設定したフォントの種類やサイズ・色は、次に変更されるまでそのまま継続されます。Wordのように改行すると標準のフォント属性にリセットされるような機能までは組み込んでいません。必要であれば、各自で考えて組み込んでみてください。

2　最初に、フォントの種類とフォントサイズ、文字色の設定機能を組み込みます。まず、フォームのLoadイベントプロシージャにコードを追加して、コンボボックスコントロールにリスト項目を設定して初期化します。

フォント名表示のコンボボックスコントロール［Combo1］に、コンピュータで利用可能なフォント名をWindowsから取得します。「Screen」オブジェクトの［FontCount］プロパティを使うと、使用しているWindowsで利用可能なフォントの数を把握できます。これを、変数「FontNum」に格納します。

```
FontNum = Screen.FontCount
```

次に、For...Nextループを使用してコンボボックスのリスト項目にフォント名を組み込みます。「Screen」オブジェクトの［Fonts］プロパティには、利用できるフォント名が格納されています。このプロパティにアクセスすると、フォント名を取得できます。［Fonts］プロパティは、引数にインデックス番号を指定する必要があります。このインデックスは、0から始まり、フォントの総数から1を差し引いた数が該当します。そこで、ループのカウンタ用変数には、変数「FontNum-1」を上限に設定すると、すべてのフォント名を取得できます。

```
For i = 0 To FontNum - 1
    FntName = Screen.Fonts(i)
```

取得したフォント名は、コンボボックスコントロールのAddItemメソッドを使って、リスト項目に追加します。

```
Combo1.AddItem FntName
```

取得したフォント名が、リッチテキストボックスコントロールに設定したフォント名と一致した場合は、そのフォント名をコンボボックスコントロールのデフォルトのフォント名として設定しておきます。

```
If FntName = RichTextBox1.Font Then
    Combo1.Text = FntName
End If
```

このプログラムでは、リッチテキストボックスコントロールのフォントを「Arial」に設定していますから、このフォント名がコンボボックスコントロールのデフォルトフォント名になります。また、このコンボボックスコントロールは、[Sorted] プロパティを「True」に設定していますので、フォント名のリストは昇順に並べられて表示されます。

3 フォント名のコンボボックスコントロールが作成できたら、今度はフォントサイズ用のコンボボックスコントロールを初期化します。これは、ただ単純にFor...Nextループを使って、数字をAddItemメソッドで追加していくだけです。Forループには「Step 2」を付け加え、フォントサイズが2つずつ増加されて表示されるようにします。

また、リッチテキストボックスコントロールに設定したフォントサイズと一致した場合は、そのフォントサイズをコンボボックスコントロールのデフォルトのフォントサイズとして設定しておきます。この方法を使用すると、現在の設定値をコンボボックスのリスト項目として表示させることができます。

```
For j = 2 To 30 Step 2
    Combo2.AddItem j
    If j = RichTextBox1.Font.Size Then
        Combo2.Text = j
    End If
Next
```

2から30までの範囲でサイズが表示される

4 コンボボックスコントロールの初期化が終了したら、実際にフォントを変更する処理を作成しましょう。リッチテキストボックスコントロールに入力された文字が選択されると、リッチテキストボックスコントロールの「Sel」で始まるプロパティを操作できるようになります。

選択された文字のフォントを変更するには、リッチテキストボックスコントロールの［SelFontName］プロパティに、変更するフォント名を設定すると、選択した文字がそのフォントに置き換えられます。そこで、コンボボックスコントロールからフォントが選択されれば、このフォント名を［SelFontName］プロパティに設定する処理を、コンボボックスコントロールの「Click」イベントプロシージャに組み込みます。

コンボボックスコントロールでクリックされたリスト項目は、コンボボックスコントロールの［Text］プロパティに格納されていますから、この値をそのままリッチテキストボックスコントロールの［SelFontName］プロパティに代入します。そして、リッチテキストボックスコントロールにフォーカスを戻しておけば、すぐに次の入力が可能になります。

```
Private Sub Combo1_Click()
    With RichTextBox1
        .SelFontName = Combo1.Text
        .SetFocus
    End With
End Sub
```

なお、リッチテキストボックスコントロールの［HideSelection］プロパティを「False」にしているため、フォーカスが他のコントロールに移っても、選択部分の反転表示はそのままになります。

ワンポイントアドバイス

リッチテキストボックスコントロールのこの機能では、いくつかの制限が発生します。

①英文を選択して日本語フォントを適用することができますが、日本語文を選択して英字フォントを適用できません。文字化けを起こすことがあります。
②リッチテキストボックスコントロールのフォント設定を日本語にすると、英字のフォント変更がうまくいかない場合があります。
③他のソフトから文字をコピーして貼り付ける場合は、英字・日本語のフォントにうまく対応できない場合があります。

5　フォントのサイズを変更する処理も、ほぼ同様の手法で行います。フォントサイズ用のコンボボックスコントロールがクリックされると、[Text] プロパティからフォント名を取得して、リッチテキストボックスコントロールの [SelFontSize] プロパティに格納します。このプロパティは、選択した文字のフォントサイズを設定します。

```
Private Sub Combo2_Click()
    With RichTextBox1
        .SelFontSize = Combo2.Text
        .SetFocus
    End With
End Sub
```

そしてもう1つ、サイズが直接コンボボックスに入力された場合の処理を作成します。フォントサイズ用のコンボボックスは、最大フォントサイズを「30」までしか設定していません。これ以上大きなサイズを使いたい場合は、直接コンボボックスコントロールのエディット部分にサイズを入力してもらうことになります。そこで、入力されたらこの数値を受け取って、リッチテキストボックスコントロールに渡す処理が必要になります。

ここでは、コンボボックスコントロールのエディット部分に数値が入力されて、Enterキーが押された時点でこの数値をリッチテキストボックスコントロールの[SelFontSize] プロパティに代入するようにします。

コンボボックスコントロールに対しいずれかのキーが押されると、コンボボックスコントロールには「KeyDown」イベントが発生します。このイベントプロシージャは引数にキーの種類を持ち、イベントが発生した時点でこの引数に押されたキーの種類が定数で格納されます。

Enterキーが押されると、引数「KeyCode」には「vbKeyReturn」(13) という定数が格納されますので、Ifステートメントでキーの種類を判断して、[SelFontSize] プロパティへの数値の転送を実行します。

```
Private Sub Combo2_KeyDown(KeyCode As Integer, Shift As_
                                            Integer)
    If KeyCode = vbKeyReturn Then
        RichTextBox1.SelFontSize = Combo2.Text
    End If
End Sub
```

これで、コンボボックスコントロールにフォントサイズが入力されてれば、選択文字列をそのサイズに変更することができます。

6

文字色の設定は、ツールバーに配置したフレームコントロールで行います。このコントロールは、現在の文字色を表示し、設定のためのコモンダイアログボックスを呼び出します。

フレームコントロールがクリックされると、Clickイベントが発生します。そこで、このイベントプロシージャにコモンダイアログボックスコントロールの「ShowColor」メソッドを実行して、[色の設定] ダイアログボックスを表示します。

ユーザーによって任意の色が選択されると、コモンダイアログボックスコントロールの [Color] プロパティに選択された色の値が格納されますから、これをリッチテキストボックスコントロールの [SelColor] プロパティに代入します。[SelColor] プロパティは、現在選択されている文字列の色を設定するプロパティです。

また、同じ値をフレームコントロールの [BackColor] プロパティに代入して、選択した色が何色であるのかを表示するようにします。

```
Private Sub Frame1_Click()
    With CommonDialog1
        .ShowColor
        Frame1.BackColor = .Color
        RichTextBox1.SelColor = .Color
    End With
End Sub
```

7 今度は、選択した文字列の表示位置を変更する処理を作成します。リッチテキストボックスコントロールの［SelAlignment］プロパティを使うと、選択した文字列の表示位置を「左揃え」「中央揃え」「右揃え」に設定できます。そこで、これらの操作を、ツールバーコントロールの3つのボタンに組み込みます。

ツールバーコントロールのButtonClickイベントプロシージャで、キー「right」「center」「left」の値のCase節に、［SelAlignment］プロパティの値の代入式を記述します。［SelAlignment］プロパティには、それぞれの表示位置を表す専用の定数を設定します。

定数	表示位置
rtfLeft	左揃え
rtfCenter	中央揃え
rtfRight	右揃え

```
Private Sub Toolbar1_ButtonClick(ByVal Button As_
                                  MSComctlLib.Button)
    Select Case Button.Key
        Case "save"
            Call FileSave

        Case "open"
            Call FileLoad

        Case "center"
            RichTextBox1.SelAlignment = rtfCenter

        Case "left"
            RichTextBox1.SelAlignment = rtfLeft

        Case "right"
            RichTextBox1.SelAlignment = rtfRight

        Case "print"
            Call Printout
    End Select

End Sub
```

新規文書
ファイル
Arial 14

表示位置
表示位置
表示位置

それぞれの表示位置

8 最後に、印刷の処理を作成します。リッチテキストボックスコントロールの「SelPrint」メソッドを使用すると、選択範囲を印刷できます。また、選択範囲を指定しないでこのメソッドを実行すると、文書全体を印刷できます。

印刷の実行は、ツールバーのボタンとメニューの2個所で行えるようにしますので、印刷処理全体を独自のプロシージャとして作成して、それぞれのコントロールで呼び出すようにしています。

印刷処理のプロシージャを「Printout」という名前で作成します。まず、このプロシージャが呼ばれると、最初にメッセージボックスを表示し、印刷の実行を確認します。メッセージボックスには、文書のファイル名を表示し、[はい]と[いいえ]の2つのボタンを持ったメッセージボックスを作成します。そして、MsgBox関数の戻り値は変数「Ret」に格納します。

```
Private Sub Printout()
    Dim Ret As Integer

    Ret = MsgBox(Form1.Caption & " を印刷しますか？", _
                 vbYesNo, "印刷の実行")
```

ユーザーによってメッセージボックスの[はい]ボタンがクリックされたら、リッチテキストボックスコントロールの「SelPrint」メソッドを実行して印刷します。

このメソッドは、引数にデバイスの「デバイスコンテキスト」を指定します。「SelPrint」メソッドは、直接プリンタを制御して印刷するのではなく、リッチテキストボックスコントロールの内容を「Printer」オブジェクトにいったん送信します。そして、「Printer」オブジェクトが実際の印刷作業を実行します。そこで、印刷のための出力先を指定するのに、「Printer」オブジェクトの[hDC]プロパティを指定します。

```
If Ret = vbYes Then
    Form1.RichTextBox1.SelPrint Printer.hDC
End If
```

「Printout」プロシージャの仕事はこれだけです。あとは、リッチテキストボックスコントロールとPrinterオブジェクトが印刷処理を実行してくれます。

印刷を実行するダイアログボックス

9 印刷処理のプロシージャが完成したら、まずツールバーの［印刷］ボタンでこのプロシージャを呼び出す処理を、ツールバーコントロールのButtonClickイベントプロシージャに加えます。［印刷］ボタンのキーの値は「print」になります。

```
Case "print"
    Call Printout
```

また、［ファイル－印刷］メニューのClickイベントプロシージャにも、この「Printout」プロシージャを呼び出す処理を記述します。これで完成です。

```
Private Sub IDM_PRINT_Click()
    Call Printout
End Sub
```

まとめ

リッチテキストボックスコントロールは、フォントの操作に一部癖はありますが、フォント属性の変更や文字位置の操作、段落の設定や印刷処理などがプロパティやメソッドで簡単に作成できるので、とても便利です。
印刷機能も装備されており、プログラムにちょっとした文書編集機能を組み込むには、使い勝手のいいコントロールになっています。

練習問題

Q1
印刷実行時に、コモンダイアログボックスの［印刷］ダイアログボックスを表示して、印刷を実行するような処理に変更してください。

【ヒント】
［印刷］ダイアログボックスを表示するには、「ShowPrinter」メソッドを実行します。また、このメソッドを実行する前に、コモンダイアログボックスコントロールの［Flags］プロパティにいくつかの設定を行う必要があります。

解答は巻末に→

Let's VBSchool!!

第3日（Lesson8～Lesson11）
便利なコントロールを使う

Visual Basic 6.0には、とても便利なコントロールが標準で組み込まれています。たとえば、スライダーコントロールは、マウスでつまみをスライドさせることで、データを連続入力できます。プログレスバーコントロールは、処理の進行を目盛の増加で表示します。そのほかにも、補足情報を表示するステータスバーコントロールや、ボタンでデータの入力や操作を行うアップダウンコントロールなどがあります。これらのコントロールをプログラムの処理に組み込んで使うと、ユーザーの操作性が向上し、使いやすいプログラムを作成できます。

今日のレッスンでは、これらの便利な４つのコントロールを、プログラムに組み込むための基本的な知識を身に付けます。

今日のレッスンで解説する項目

- ❑ スライダーコントロールのプロパティ設定
- ❑ スライダーコントロールとラベルコントロールの連動処理
- ❑ スライダーコントロールのイベント処理
- ❑ プログレスバーコントロールの機能
- ❑ プログレスバーコントロールのプロパティ設定
- ❑ プログレスバーを使った状態表示処理の作成
- ❑ ステータスバーコントロールのプロパティ設定
- ❑ パネルへの表示方法
- ❑ ステータスバーの活用テクニック
- ❑ パネルの追加とスタイル設定
- ❑ アップダウンコントロールのプロパティ設定
- ❑ 連動コントロールとの同期設定
- ❑ イベントプロシージャの使い方

Lesson8.vbp
VB6CD2
Lesson08

スライダーコントロールを使う

スライダーコントロールは、目盛とバーとつまみボタンが組み合わされたコントロールです。マウスでボタンをスライドさせることで、数値の入力などを連続して行うことができます。このレッスンでは、簡単なプログラムを作成して、スライダーコントロールを使うためのテクニックを紹介します。

今回作成するプログラム

Form1

16

スライダのつまみをドラッグすると…

ラベルの幅が変化する

POINT

スライダーコントロールを操作すると、ラベルコントロールの幅が変化するプログラムです。ラベルコントロールには、そのときのスライダーコントロールの値が表示され、現在のボタンの位置を把握できるようにしています。

スライダーコントロールを操作する上で必要なプロパティの設定を身に付けてください。

コントロールとプロパティの設定

※新しいプロジェクトファイル（標準EXE）を用意します。

1 ツールパレットに「Microsoft Windows Common Controls 6.0」コンポーネントを追加する

ツールパレットにコモンコントロールが追加される

2 ツールパレットの［Label］ A を使って、ラベルコントロールを配置する

Form1

Label1

1 斜めにドラッグして配置

3 ［BorderStyle］プロパティを「1 －実線」に設定する

プロパティ - Label1

(ｵﾌﾞｼﾞｪｸﾄ名)	Label1
Alignment	0 - 左揃え
Appearance	1 - 3D
AutoSize	False
BackColor	&H8000000F&
BackStyle	1 - 不透明
BorderStyle	1 - 実線
Caption	0 - なし / 1 - 実線
DataField	
DataFormat	

1 ここをクリック

2 ここをクリック

3 ここをクリック

コンポーネントを追加する

コンポーネントを追加する方法については、「Lesson5 ツールバーコントロールの組み込み」を参照してください。

選択肢の一覧を表示する

［BorderStyle］プロパティの右側の欄の右端にある下向き三角をクリックすると、選択肢の一覧が表示されます。

4 [Width] プロパティを「100」に設定する

ToolTipText	
Top	240
UseMnemonic	True
Visible	True
WhatsThisHelpID	0
Width	100
WordWrap	False

1 「100」に設定

5 [Caption] プロパティの値を削除する

Appearance	1 - 3D
AutoSize	False
BackColor	&H8000000F&
BackStyle	1 - 不透明
BorderStyle	1 - 実線
Caption	
DataField	
DataFormat	

1 値を削除

6 ツールパレットの [Slider] を使って、スライダーコントロールを配置する

Form1

1 斜めにドラッグして配置

7 プロパティページを開く

プロパティ - Slider1

Slider1 Slider

全体 | 項目別

(オブジェクト名)	Slider1
(バージョン情報)	
(プロパティページ)	...
BorderStyle	0 - ccNone

1 ここをクリック

2 ここをクリック

[...] ボタン

[...] ボタンは、そのプロパティの詳細な指定を行うダイアログボックスが用意されていることを示しています。

8 ［最小値］を「1」に、［最大値］を「20」に設定する

プロパティページ

全般 | 描画スタイル | ピクチャ

最小値(I): 1　☑ イベント認識(N)
最大値(X): 20　☐ 選択範囲(R)
最小移動量/増分(M): 1　選択範囲の先頭(S): 1
最大移動量/増分(L): 5　選択範囲の長さ(E): 0
OLE ドロップ モード(O): 0 - ccOLEDropNone

OK　キャンセル　適用(A)　ヘルプ

1 「1」を設定

2 「20」を設定

コードの記述

Form1:Slider1-Change

Lesson8 - Form1 (コード)

Slider1　Change

```
Private Sub Slider1_Change()
    With Label1
        .Width = Slider1.Value * 100
    End With
    Label1.Caption = Slider1.Value
End Sub
```

解説

1　スライダーコントロールは、スクロールバーコントロールとよく似たプロパティを持っています。

まず、［Max］（最大値）と［Min］（最小値）プロパティで、スライダーコントロールの可動範囲を設定します。［SmallChange］（最小移動量/増分）と［LargeChange］（最大移動量/増分）プロパティは、スライダーの実際の移動量を設定するプロパティです。［SmallChange］プロパティは、スライドつまみをマウスでドラッグした場合、またはキーボードの左右の矢印キーを押したときに移動する量を設定します。［LargeChange］プロパティは、スライダーのバーの部分をマウスでクリックしたとき、またはキーボードのPageUpキーまたはPageDownキーを押したときに動作する移動量を設定します。［Orientation］（配置方向）プロパティ

を使用すると、スライダーコントロールの水平・垂直方向を設定できます。
これらは、プロパティページでまとめて設定することができます。

プロパティページで配置方向を設定

垂直スライダーコントロール

［TickStyle］プロパティは、スライダーコントロールに付ける目盛りの位置を設定します。また、［TickFrequency］プロパティは、目盛りの増分を設定します。
これらのプロパティを組み合わせて、使用する処理にあったスライダーコントロールの形状と動作範囲を設定します。

2

このプログラムは、スライダーコントロールを操作すると、フォームに配置したラベルコントロールの幅が変化するプログラムです。
スライダーコントロールは、［Max］プロパティを「20」に、［Min］プロパティを「1」に設定しているので、この範囲で動作します。スライダーコントロールを操作すると、このコントロールには「Change」イベントが発生します。これは、マウスとキーボードのどちらで操作しても、このイベントが発生します。そこで、スライダーコントロールの操作に合わせた処理を行うには、このイベントプロシージャに処理を組み立てます。
スライダーコントロールの動作量は、［Value］プロパティに格納されます。値は長整数型（Long）です。この値を取り出せば、現在のスライダーコントロールの動作量を把握できますから、この値をラベルコントロールの［Width］プロパティに代入すると、スライダーコントロールの動作に連動して、ラベルコントロールの幅が変化します。
また、［Value］プロパティの値をラベルコントロールの［Caption］プロパティに代入して、その数値がわかるようにします。

```
Private Sub Slider1_Change()
    With Label1
        .Width = Slider1.Value * 100
    End With
    Label1.Caption = Slider1.Value
End Sub
```

スライダーを動かすと...

ラベルコントロールの幅が変化する

まとめ

スライダーコントロールは、ユーザーの入力操作を助ける便利なコントロールです。単独で使うよりも、マルチメディアコントロールやテキストボックスコントロールなどの、他のコントロールと組み合わせて使うと面白い効果が得られます。

練習問題

Q1　イメージコントロールとスライダーコントロールを組み合わせて、スライダーコントロールを操作するとイメージコントロールが左右に動くプログラムを作成してください。

スライダーコントロールの動作に合わせてイメージが移動する

解答は巻末に→

Lesson 9

第３日
２時限目

便利なコントロールを使う

Lesson9.vbp
VB6CD2
Lesson09

プログレスバーコントロールを使う

プログレスバーコントロールは、横長の空欄の中を四角いブロックで塗りつぶすことで、処理の進行状況を示すコントロールです。このコントロールをプログラムに組み込むことで、処理に時間がかかる場合はその状況をユーザーに伝えることができます。

今回作成するプログラム

Form1

実行すると…

Form1

１秒ごとにブロックが増えていく

POINT

プログレスバーとタイマーコントロールを組み合わせて、１秒ごとにプログレスバーで進行状況を表示します。

また、プログレスバーといえどもイベントを持っていますので、このイベントにタイマーコントロールを停止する処理を組み込みます。

コントロールとプロパティの設定

※新しいプロジェクトファイル（標準EXE）を用意します。

1 ツールパレットに「Microsoft Windows Common Controls 6.0」コンポーネントを追加する

ツールパレットにコモンコントロールが追加される

2 ツールパレットの［ProgressBar］ を使って、フォームにプログレスバーコントロールを配置する

Form1

1 斜めにドラッグして配置

3 サイズ変更ハンドルをドラッグして、黒四角が同じ大きさで10個揃うようにサイズ調整する

Form1

1 ここをドラッグ

コンポーネントを追加する

コンポーネントを追加する方法については、「Lesson5 ツールバーコントロールの組み込み」を参照してください。

4 [Max] プロパティを「10」に設定する

プロパティ - ProgressBar1	
ProgressBar1	ProgressBar
Align	0 - vbAlignNone
Appearance	1 - cc3D
BorderStyle	0 - ccNone
DragIcon	(なし)
DragMode	0 - vbManual
Enabled	True
Height	660
Index	
Left	210
Max	10
Min	0

1 「10」に設定

5 ツールパレットの [Timer] を使って、タイマーコントロールを配置する

Form1

1 斜めにドラッグして配置

6 [Interval] プロパティを「1000」に設定する

プロパティ - Timer1	
Timer1	Timer
(オブジェクト名)	Timer1
Enabled	True
Index	
Interval	1000
Left	3630
Tag	
Top	2205

1 「1000」に設定

コードの記述

Form1:(General)-(Declarations)

Lesson9 - Form1 (コード)

(General) ｜ (Declarations)

```
Dim i As Integer
```

Form1:Form-Load

Lesson9 - Form1 (コード)

Form ｜ Load

```
Private Sub Form_Load()
    i = 1
End Sub
```

Form1:ProgressBar1-Click

Lesson9 - Form1 (コード)

ProgressBar1 ｜ Click

```
Private Sub ProgressBar1_Click()
    Timer1.Interval = 0
    MsgBox "タイマーを停止しています。"
    Timer1.Interval = 1000
End Sub
```

Form1:Timer1-Timer

Lesson9 - Form1 (コード)

Timer1 ｜ Timer

```
Private Sub Timer1_Timer()
    With ProgressBar1
        .Value = i
        If i = .Max Then
            i = .Min
        End If
    End With
    i = i + 1
End Sub
```

解説

1 プログレスバーコントロールは、スライダーコントロールと同様に［Max］と［Min］プロパティで可動範囲を設定します。この可動範囲とコントロールの幅によって自動的に計算され、コントロールの中の黒い四角が動作していきます。この黒四角の現在のポジションは、［Value］プロパティによって決まります。そこで、このプログラムでは、タイマーコントロールを使って、1秒ごとにプログレスバーコントロールの［Value］プロパティの値を1つずつ増加し、処理の状況を刻一刻と伝えるようにしています。

2 最初に、プログレスバーコントロールの［Value］プロパティに設定する値を格納する変数「i」を、モジュールレベルの変数で宣言します。

```
Dim i As Integer
```

そして、フォームのLoadイベントプロシージャで初期化します。

```
Private Sub Form_Load()
    i = 1
End Sub
```

3 タイマーコントロールの「Timer」イベントプロシージャで、プログレスバーの動作処理を作成します。このイベントプロシージャは、［Interval］プロパティを「1000」に設定しているため、1秒ごとに呼び出されて処理が実行されます。

まず、プログレスバーコントロールの［Value］プロパティに、変数「i」の値を代入します。もし、「i」の値がプログレスバーコントロールの［Max］プロパティの値を超えてしまうとプログラムはエラーになってしまうので、そのときは［Min］プロパティの値を変数「i」に代入してリセットします。そして、この変数「i」を1つ増加してプロシージャを終了します。

このプロシージャは、1秒ごとに実行され、変数「i」の値が1つ増加してプログレスバーコントロールの［Value］プロパティに代入されます。結果として、プログレスバーコントロールは、1秒ごとに黒い四角を表示します。

```
Private Sub Timer1_Timer()
    With ProgressBar1
        .Value = i
        If i = .Max Then
            i = .Min
        End If
    End With
    i = i + 1
End Sub
```

4　プログレスバーコントロールは、他のコントロールと同じ数のイベントを持っています。そこで、このコントロールがクリックされると、タイマーコントロールが停止し、また再起動する処理を組み込んでみます。プログレスバーコントロールがマウスでクリックされると、Clickイベントプロシージャが発生します。そこで、このイベントプロシージャに、タイマーコントロールの［Interval］プロパティを「0」にする処理を組み込みます。これでタイマーコントロールは動作を停止しますから、プログレスバーコントロールの動作も停止します。

次にメッセージボックスを表示し、［OK］ボタンがクリックされると再度［Interval］プロパティに値「1000」を設定して動作を再開します。プログレスバーコントロールも、これに連動して動作を開始します。

```
Private Sub ProgressBar1_Click()
    Timer1.Interval = 0
    MsgBox "タイマーを停止しています。"
    Timer1.Interval = 1000
End Sub
```

まとめ

プログラムの内部動作が見えないユーザーにとっては、プログラムが現在何をしているのかを表示するプログレスバーは大切なインタフェースです。プログレスバーコントロールのサイズを、動作範囲に合わせて上手に調整するのがポイントです。

練習問題

Q1　「Windows」フォルダ内にあるファイルの総数を調べるプログラムを作成してください。進行状況はプログレスバーで表示してください。Dir関数とDo...Whileループを組み合わせてファイルの個数を調べます。

Form1

検索

Q&A-09

ファイルの総数は 325個です

OK

ファイルの総数を表示するプログラム。進行状況をプログレスバーで表示する

解答は巻末に→

Lesson 10

第３日
３時限目

便利な
コントロール
を使う

Lesson10.vbp
VB6CD2
Lesson10

ステータスバーコントロールを使う

ステータスバーコントロールも、プログラム動作時のさまざまな情報を表示する便利なコントロールです。Windowsユーザーには、もうすっかりおなじみのコントロールで、ウィンドウの最下部にいろいろな文字を表示します。このレッスンでは、ステータスバーコントロールのプログラムデザイン時の操作と、プログラム実行時の操作方法を解説します。

今回作成するプログラム

Form1
Command1
ボタンやコントロールをクリックすると...
ステータスバーにそれらに関する情報が表示される
ボタンをクリック
イメージコントロール
10:34
さらにステータスバーに時計を組み込む

POINT

いくつかのコントロールとステータスバーコントロールを組み合わせて、ステータスバーに情報を表示します。

ステータスバーコントロールは３つのパネルを持ち、１つはフォーム上のコントロールをマウスでポイントすると、そのコントロール名がステータスバーに表示されます。もう１つはコマンドボタンをクリックすると、その回数をパネルに表示します。最後のパネルは、アイコンを組み込み、現在の時刻を表示します。

コントロールとプロパティの設定

※新しいプロジェクトファイル（標準EXE）を用意します。また、付録CD-ROMに収録されている「material」フォルダの「FLOPPY_1.ICO」と「CLOCK8.ICO」を使用します。

1 ツールパレットに「Microsoft Windows Common Controls 6.0」コンポーネントを追加する

ツールパレットにコモンコントロールが追加される

2 ツールパレットの［CommandButton］を使って、コマンドボタンコントロールを配置する

Form1

Command1

1 斜めにドラッグして配置

3 ツールパレットの［Image］を使って、イメージコントロールを配置する

Form1

Command1

1 斜めにドラッグして配置

コンポーネントを追加する

コンポーネントを追加する方法については、「Lesson5 ツールバーコントロールの組み込み」を参照してください。

4 [Picture] プロパティの [...] ボタンをクリックし、「FLOPPY_1.ICO」を指定する

1 ここをクリック

2 ここをクリック

5 ツールパレットの [StatusBar] を使って、ステータスバーコントロールを配置する

1 斜めにドラッグして配置

6 [プロパティページ] の [パネル] タブで、[テキスト] に「ボタンをクリック」と入力する

1 ここに入力

[...] ボタン

[...] ボタンは、そのプロパティの詳細な指定を行うダイアログボックスが用意されていることを示しています。

ステータスバーコントロール

ボタンをクリックしてフォーム上をドラッグすると、ステータスバーコントロールがフォーム下部の左右幅いっぱいに自動的に配置されます。

[プロパティページ] ダイアログ

[プロパティページ] ダイアログボックスを表示するには、ステータスバーコントロールが選択されている状態で、プロパティウィンドウの [(プロパティページ)] をクリックし、右欄の [...] ボタンをクリックします。

7 ［キー］に「one」と入力し、［最小幅］を「1440」に設定する

1 「one」と入力

2 「1440」に設定

8 ［パネルの挿入］をクリックし、［キー］に「two」と入力し、［最小幅］に「1700」を設定する

1 ここをクリック

2 「two」と入力

3 「1700」に設定

9 ［パネルの挿入］をクリックし、［キー］に「three」を入力する

1 ここをクリック

2 「three」と入力

10 ［スタイル］で［5 − sbrTime］を選択し、［ピクチャ］の［参照］ボタンをクリックする

「5-sbrTime」を選択

ここをクリック

［スタイル］リストボックス

［スタイル］リストボックスの右端にある下向き三角をクリックすると、選択肢の一覧が表示されます。その一覧で「5 − sbrTime」を選択します。

［ピクチャ］の［参照］ボタン

［参照］ボタンをクリックすると、ピクチャを選択する［図の選択］ダイアログボックスが表示されます。

11 「CLOCK8.ICO」を選択し、［開く］をクリックする

ここをクリック

ここをクリック

作業を終了する

［開く］ボタンをクリックすると［プロパティページ］ダイアログボックスに戻るので、［OK］ボタンをクリックして作業を終了してください。

12 完成したフォーム

コードの記述

Form1:(General)-(Declarations)

Lesson10 - Form1 (ｺｰﾄﾞ)
(General) / (Declarations)

```
Dim i As Integer
```

Form1:Command1-Click

Lesson10 - Form1 (ｺｰﾄﾞ)
Command1 / Click

```
Private Sub Command1_Click()
    i = i + 1
    StatusBar1.Panels(1).Text = "Push " & i
End Sub
```

Form1:Command1-MouseMove

Lesson10 - Form1 (ｺｰﾄﾞ)
Command1 / MouseMove

```
Private Sub Command1_MouseMove(Button As Integer, _
                    Shift As Integer, X As Single, Y As Single)
    StatusBar1.Panels(2).Text = "コマンドボタン"
End Sub
```

Form1:Form-Load

Lesson10 - Form1 (ｺｰﾄﾞ)
Form / Load

```
Private Sub Form_Load()
    i = 0
End Sub
```

Form1:Form-MouseMove

Lesson10 - Form1 (ｺｰﾄﾞ)
Form / MouseMove

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, _
    X As Single, Y As Single)
    StatusBar1.Panels("two").Text = "フォーム"
End Sub
```

Form1:Image1-MouseMove

Lesson10 - Form1 (コード)

Image1 | MouseMove

```
Private Sub Image1_MouseMove(Button As Integer, _
    Shift As Integer, X As Single, Y As Single)
    StatusBar1.Panels(2).Text = "イメージコントロール"
End Sub
```

Form1:StatusBar1-MouseMove

Lesson10 - Form1 (コード)

StatusBar1 | MouseMove

```
Private Sub StatusBar1_MouseMove(Button As Integer, Shift As _
    Integer, X As Single, Y As Single)
    StatusBar1.Panels(2).Text = "ステータスバー"
End Sub
```

解 説

1 ステータスバーコントロールは、コントロール内をいくつかの表示エリアに区切り、それぞれ別々の内容を表示できるようになっています。このエリアを「パネル」と呼びます。1つのステータスバーコントロールで、合計16個までのパネルを設定できるようになっています。

プロパティページを使用すると、このパネルをコントロールに追加でき、またパネルのサイズやテキスト・イメージの設定が行えます。フォームに配置したばかりのステータスバーコントロールは、1つしかパネルを持っていませんので、デザイン時にプロパティシートを使って必要な数だけパネルを追加します。

プロパティページ

全般 | パネル | フォント | ピクチャ

インデックス(I): 1　パネルの挿入(N)　パネルの削除(R)

テキスト(T): ボタンをクリック

ツール ヒント(X):

キー(K): one　最小幅(W): 1440.00

タグ(G):　実際の幅(C): 1440.00

配置(M): 0 - sbrLeft　ピクチャ

スタイル(S): 0 - sbrText　参照(O)...

立体枠(B): 1 - sbrInset　ピクチャなし(P)

サイズの自動設定(U): 0 - sbrNoAutoSize　☑ イベント認識(E)　☑ 表示(V)

OK　キャンセル　適用(A)　ヘルプ

プロパティページでパネルを追加する

また、プロパティウィンドウにある［Align］プロパティを使用すると、フォーム内のステータスバーコントロールの表示位置を設定できます。デフォルトではフォームの最下部にくるような設定になっていますが、このプロパティを使えばフォームの上下左右にコントロールを配置できます。

［Align］プロパティで表示位置を指定すると…

フォームの上端に配置される

2　設定したパネルは、［サイズの自動設定］でパネルのサイズ調整ができます。ここには定数を設定します。

定数	設定
0 − sbrNoAutoSize	（既定値）常に Width プロパティに設定された値になります。
1 − sbrSpring	フォームの大きさにあわせて自動的にステータスバーコントロールの幅が設定されると、この設定を持つすべてのパネルでコントロール幅を等分割してパネルの大きさを変更します。
2 − sbrsbrContents	Panel に設定された内容に合わせて自動的に大きさが変更されます。ただし、パネルの幅が MinWidth プロパティに設定された値より小さくなることはありません。

［サイズの自動設定］が「0 − sbrNoAutoSize」になっている場合は、［最小幅］で設定したサイズがパネルの幅になります。

Form1

Command1

ボタンをクリック　10:53

「1 − sbrSpring」を設定すると、スライダーコントロールの幅いっぱいにパネルの幅が設定され余白がなくなる

3 ［スタイル］を選ぶと、あらかじめいくつかの機能をパネルに組み込むことができます。これらは定数で設定するようになっています。たとえば、［スタイル］に定数「sbrTime」を設定すると、パネルには現在の時刻が表示されます。

定数	値	内容
sbrText	0	（既定値）文字列とビットマップ。文字列は［Text］プロパティで設定します。
sbrCaps	1	Caps Lockキーがオンの場合、"CAPS"の文字が太字で表示さ、れオフの場合は淡色表示されます。
sbrNum	2	Num Lockキーがオンの場合、"NUM"の文字が太字で表示され、オフの場合は淡色表示されます。
sbrIns	3	Insキーがオンの場合、"INS"の文字が太字で表示され、オフの場合は淡色表示されます。
sbrScrl	4	Scroll Lockキーがオンの場合、"SCRL"の文字が太字で表示され、オフの場合は淡色表示されます。
sbrTime	5	現在の時刻をシステムで設定された形式で表示します。
sbrDate	6	現在の日付をシステムで設定された形式で表示します。
sbrKana	7	カナキーがオンの場合、"カナ"の文字が太字で表示され、オフの場合は淡色表示されます。

各スタイル

テキスト　CAPS　NUM　INS　SCRL　11:04　99/09/05　カナ

また、［ピクチャ］にイメージファイルを設定すると、パネル内にイメージを表示できます。組み込めるイメージは、ビットマップ（.BMP）とアイコン（.ICO）形式のイメージファイルになります。

設定したイメージファイル

4　ステータスバーコントロールにパネルを設定すると、パネルは「Panel」オブジェクトとして扱われます。複数のパネルがある場合は、「Panels」コレクションとして扱われます。個々のパネルのプロパティ設定については、この「Panel」オブジェクトのプロパティとメソッドを操作します。

複数のパネルを設定した場合、それぞれのパネルへのアクセスは、オブジェクトに「Panels()」コレクションを指定します。()の中には、［Index］プロパティまたは［Key］プロパティの値を指定することで、個々のパネルを識別します。たとえば次の記述は、このプログラムに組み込んだステータスバーコントロールの、最初のパネルに文字を表示します。

```
StatusBar1.Panels(1).Text = "Hello！"

StatusBar1.Panels("one").Text = "Hello！"
```

最初の記述は、( ) の中にプロパティページで設定したインデックスの値を指定しています。2つめの記述は、キーを文字列として指定しています。いずれの方法でも、同じパネルを操作対象にすることができます。

プロパティページでのインデックスとキーの設定

5 では、順番にコードを作成していきましょう。まず最初は、一番目のパネルに、コマンドボタンがクリックされた回数を表示する処理です。

クリックされたボタンの回数を表示

フォームのコードウィンドウの、宣言セクションに、変数「i」を宣言します。この変数に、コマンドボタンがクリックされた回数をカウントして格納します。宣言したら、フォームのLoadイベントプロシージャで、初期化しておきます。

```
Dim i As Integer

Private Sub Form_Load()
    i = 0
End Sub
```

コマンドボタンのClickイベントプロシージャに、コマンドボタンがクリックさ

れると変数「i」の値を１つ増やす処理を作成します。そして、最初のパネル「Panels(1)」の［Text］プロパティに、この変数と文字列を組みあわせて代入します。

これで、コマンドボタンがクリックされるたびに、変数「i」の値が増加しステータスバーコントロールの左端のパネルに表示されます。

```
Private Sub Command1_Click()
    i = i + 1
    StatusBar1.Panels(1).Text = "Push " & i
End Sub
```

パネルの個々のプロパティを操作する場合は、このように操作対象のパネルを指定して、パネルのプロパティを操作します。

！ ワンポイントアドバイス

ステータスバーコントロールが持っているプロパティやメソッドは、ステータスバーコントロール全体に関するプロパティです。ステータスバーコントロールそのものを操作したい場合は、このプロパティやメソッドを操作します。一方、パネルに対して操作を行いたい場合は、パネルのプロパティやメソッドを操作します。その場合は、「Panels()」コレクションを操作対象のオブジェクトに指定し、プロパティやメソッドを使うようにします。

6　次は、フォーム上のコントロールにマウスのポインタが重なったときに、そのものの名称をステータスバーコントロールの2番目のパネルに表示する処理を作成します。これは、それぞれのコントロールやフォームの「MouseMove」イベントに、ステータスバーコントロールのパネルに情報を表示する処理を組み込んでいきます。

フォームには、フォームとコマンドボタン、イメージ、ステータスバーの4つのオブジェクトがあります。それぞれ、マウスのポインタがオブジェクト上で動作すると、それぞれのオブジェクトに「MouseMove」イベントが発生します。そこで、このイベントプロシージャの中で、ステータスバーコントロールの2番目のパネルの［Text］プロパティに、文字列を設定する処理を行います。

なお、2番目のパネルへのアクセスに、「Panels(2)」と「Panels("two")」という、インデックスとキーの値を使った方法を使用しています。どちらの方法も、結果として同じパネルを参照することになります。

```
Private Sub Command1_MouseMove(Button As Integer, _
             Shift As Integer, X As Single, Y As Single)
    StatusBar1.Panels(2).Text = "コマンドボタン"
End Sub
```

```
Private Sub Form_MouseMove(Button As Integer, _
            Shift As Integer, X As Single, Y As Single)
    StatusBar1.Panels("two").Text = "フォーム"
End Sub

Private Sub Image1_MouseMove(Button As Integer, _
            Shift As Integer, X As Single, Y As Single)
    StatusBar1.Panels(2).Text = "イメージコントロール"
End Sub

Private Sub StatusBar1_MouseMove(Button As Integer, _
            Shift As Integer, X As Single, Y As Single)
    StatusBar1.Panels(2).Text = "ステータスバー"
End Sub
```

Form1

Command1

ボタンをクリック | コマンドボタン | 19:49

マウスの動く対象物によって、ステータスバーの表示が変化する

Form1

Command1

ボタンをクリック | イメージコントロール | 19:50

表示が「イメージコントロール」になった

まとめ

ステータスバーコントロールを使うと、プログラムの操作方法や処理の内容を事前にユーザーに知らせることができます。このことは、ユーザーによりわかりやすい操作方法を提供することになります。また、これから行おうとする処理をユーザーに前もって説明することができます。

パネルは、「Panels」コレクションを対象オブジェクトに指定して、Add メソッドを実行すると、プログラム実行時にパネルの増減を行うことができます。プログラムの動作状況によってステータスバーの項目を変化させたい場合に有効です。最大 16 個にまで分割できます。ただし、あまり表示を多用すると逆に画面がわずらわしくなるので、適当に工夫しましょう。

特に、ツールバーコントロールなどを連動して使用すると便利なコントロールです。

練習問題

Q1　コマンドボタンをクリックすると、ステータスバーコントロールのパネルに、プログレスバーのように進行状況が表示されるプログラムを作成してください。

【ヒント】

ちょっとトリッキーなテクニックです。文字の■をうまく使って、プログレスバーのように見せかけます。

解答は巻末に→

Lesson 11

第３日 ４時限目

便利なコントロールを使う

Lesson11.vbp

VB6CD2

Lesson11

アップダウンコントロールを使う

アップダウンコントロールは、スクロールバーコントロールのように２つのボタンを持ったコントロールです。ボタンをクリックすることで数値データなどを増減する機能を持っています。このコントロールの特徴は、他のコントロールと連動して動作させることができる、「連動コントロール」の機能を持っていることです。連動コントロールは、特にコードを書かなくても、連動するコントロールを設定するだけで、自動的にプロパティ値のやり取りを行い、コントロール同士を連動して動作させる機能です。このレッスンでは、アップダウンコントロールとテキストボックスコントロールやラベルコントロールを組み合わせて、便利な入力操作を行うテクニックを紹介します。

今回作成するプログラム

Form1

10

Label1

Label2

ボタンをクリックすると、数値が変わる

ボタンをクリックすると、ボックスの幅が変わる

ボタンをクリックすると、ボックスの色が変わる

POINT

アップダウンコントロールを使った、３つのコントロールの組み合わせを作成します。１つは、テキストボックスコントロールにアップダウンコントロールを連結させます。アップダウンコントロールを操作すると数値がテキストボックスに表示されます。

もう１つは、ラベルコントロールと連結させます。アップダウンコントロールを操作すると、ラベルコントロールの幅が変化します。

３つめは、シェイプコントロールにアップダウンコントロールを連結させます。シェイプコントロールの［FillColor］プロパティに連結させ、アップダウンコントロールでシェイプの色を連続的に変化させます。

コントロールとプロパティの設定

※新しいプロジェクトファイル（標準EXE）を用意します。

1 ツールパレットに「Microsoft Windows Common Comtrols-2 6.0」コンポーネントを追加する

ツールパレットにコモンコントロールが追加される

2 ツールパレットの［TextBox］ を使って、テキストボックスコントロールを配置する

Form1

Text1

1 斜めにドラッグして配置

3 ツールパレットの［UpDown］ を使って、アップダウンコントロールを配置する

Form1

Text1

1 斜めにドラッグして配置

コンポーネントを追加する

コンポーネントを追加する方法については、「Lesson5 ツールバーコントロールの組み込み」を参照してください。

[プロパティページ] ダイアログ

[プロパティページ] ダイアログボックスを表示するには、アップダウンコントロールが選択されている状態で、プロパティウィンドウの [(プロパティページ)] をクリックし、右欄の [...] ボタンをクリックします。

4 [プロパティページ] ダイアログボックスを表示し、[連動] タブをクリックする

ここをクリック

[自動設定] チェックボックス

このチェックボックスをオンにすると、[連動コントロール] ボックスにテキストボックスコントロールの名前が表示されます。

5 [自動設定] チェックボックスをオンにする

ここをチェック

ここにテキストコントロールの名前が表示される

[連動プロパティ] リストボックス

[連動プロパティ] リストボックスの右端にある下向き三角をクリックすると、選択肢の一覧が表示されます。その一覧で「Text」を選択します。

6 [連動プロパティ] リストボックスで [Text] プロパティを選択する

ここをクリック

ここをクリック

7 [スクロール] タブをクリックし、[最小] を「1」に「最大] を「100」に設定し、[OK] をクリックする

プロパティページ
全般 | 連動 | スクロール
スクロールの範囲
値(V):　最小(N):　最大(X)
0　1　100　□ 折り返し(W)
スクロールの割合
スクロール幅(I):
1
OK　キャンセル　適用(A)　ヘルプ

1 ここをクリック
2 「100」に設定
3 ここをクリック

8 ツールパレットの [Label] A を使って、ラベルコントロールを配置する

Form1
Text1
Label1

1 斜めにドラッグして配置

9 [BorderStyle] プロパティを「1 －実線」に設定する

プロパティ - Label1
Label1 Label
全体 | 項目別

(オブジェクト名)	Label1
Alignment	0 - 左揃え
Appearance	1 - 3D
AutoSize	False
BackColor	&H8000000F&
BackStyle	1 - 不透明
BorderStyle	1 - 実線
Caption	0 - なし 1 - 実線
DataField	
DataFormat	
DataMember	

1 ここをクリック
2 ここをクリック
3 ここをクリック

10 ツールパレットの［UpDown］ を使って、アップダウンコントロールを配置する

11 ［プロパティページ］の［連動］タブで［自動設定］チェックボックスをオンにする

12 ［連動プロパティ］リストボックスで［Width］を選択する

［プロパティページ］ダイアログ

［プロパティページ］ダイアログボックスを表示するには、アップダウンコントロールが選択されている状態で、プロパティウィンドウの［(プロパティページ)］をクリックし、右欄の［...］ボタンをクリックします。

［自動設定］チェックボックス

このチェックボックスをオンにすると、［連動コントロール］ボックスにラベルコントロールの名前が表示されます。

［連動プロパティ］リストボックス

［連動プロパティ］リストボックスの右端にある下向き三角をクリックすると、選択肢の一覧が表示されます。その一覧で「Width」を選択します。

13 ［スクロール］タブをクリックして以下の項目に値を設定する

ボックス	値
［最小］	1000
［最大］	3000
［スクロール幅］	50

プロパティページ
全般　連動　スクロール
スクロールの範囲
値(V): 0　最小(N): 1000　最大(X): 3000　折り返し(W)
スクロールの割合
スクロール幅(I): 50
OK　キャンセル　適用(A)　ヘルプ

これらのボックスの値を設定する

14 アップダウンコントロールの位置を移動する

Form1
Text1
Label1

1 ドラッグして移動

15 ラベルコントロールの［Width］プロパティを「1000」に設定する

MouseIcon	(なし)
MousePointer	0 - 既定値
OLEDropMode	0 - なし
RightToLeft	False
TabIndex	2
Tag	
ToolTipText	
Top	375
UseMnemonic	True
Visible	True
WhatsThisHelpID	0
Width	1000
WordWrap	False

1 「1000」を入力

ラベルを選択

この操作を行う前に、ラベルコントロールをクリックして選択します。

16 ツールパレットの［Shape］ を使って、シェイプコントロールを配置する

17 ［FillStyle］プロパティを［0－塗りつぶし］に設定する

18 ツールパレットの［UpDown］ を使って、アップダウンコントロールを配置する

19 ［Orientation］プロパティを「1 － cc2OrientationHorizontal」に設定する

プロパティ － UpDown3
UpDown3 UpDown
全体 | 項目別

OLEDropMode	0 - cc2OLEDropNone
Orientation	cc2OrientationHorizontal
SyncBuddy	0 - cc2OrientationVertical
TabIndex	1 - cc2OrientationHorizontal
TabStop	True
Tag	
ToolTipText	
Top	2340
Value	0
Visible	True

1 ここをクリック
2 ここをクリック
3 ここをクリック

20 ［プロパティページ］の［連動］タブで［連動コントロール］に「Shape1」と入力する

プロパティ ページ
全般 | 連動 | スクロール
連動コントロール(B): Shape1
□ 自動設定(D)　□ 同期(S)
連動プロパティ(P): (なし)
OK | キャンセル | 適用(A) | ヘルプ

1 「Shape1」と入力

［プロパティページ］ダイアログ

［プロパティページ］ダイアログボックスを表示するには、アップダウンコントロールが選択されている状態で、プロパティウィンドウの［(プロパティページ)］をクリックし、右欄の［...］ボタンをクリックします。

21 ［連動プロパティ］リストボックスで［FillColor］を選択する

プロパティ ページ
全般 | 連動 | スクロール
連動コントロール(B): Shape1
□ 自動設定(D)　☑ 同期(S)
連動プロパティ(P): FillColor
Tag
Shape
DrawMode
BorderStyle
BorderWidth
FillColor
BackStyle
FillStyle
OK | キャンセル | 適用(A) | ヘルプ

1 ここをクリック
2 ここをクリック

［連動プロパティ］リストボックス

［連動プロパティ］リストボックスの右端にある下向き三角をクリックすると、選択肢の一覧が表示されます。その一覧で「FillColor」を選択します。

22 [スクロール] タブをクリックして以下の項目に値を設定する

ボックス	値
[最小]	0
[最大]	255
[スクロール幅]	5

プロパティページ

全般　連動　スクロール

スクロールの範囲
値(V):　最小(N): 0　最大(X): 255　折り返し(W)

スクロールの割合
スクロール幅(I): 5

OK　キャンセル　適用(A)　ヘルプ

これらのボックスの値を設定する

23 ツールパレットの [Label] A を使って、ラベルコントロールを配置する

Form1

Text1　Label1　Label2

1 斜めにドラッグして配置

24 [BorderStyle] プロパティを [1 －実線] に設定する

Label2 Label	
(ｵﾌﾞｼﾞｪｸﾄ名)	Label2
Alignment	0 - 左揃え
Appearance	1 - 3D
AutoSize	False
BackColor	&H8000000F&
BackStyle	1 - 不透明
BorderStyle	1 - 実線
Caption	0 - なし / 1 - 実線
DataField	

1 ここをクリック

2 ここをクリック

3 ここをクリック

25 完成したフォーム

Form1

Text1　Label1　Label2

コードの記述

Form1:UpDown3-Change

Lesson11 - Form1 (ｺｰﾄﾞ)

UpDown3　Change

```
Private Sub UpDown3_Change()
    Label2.Caption = UpDown3.Value
End Sub
```

解 説

1 アップダウンコントロールは、他のコントロールと連動して動作する機能を持っています。「連動」とは、片方のコントロールを動作させると自動的にもう一方のコントロールが動作することです。通常、このような動作はコントロールのイベントプロシージャにコードを作成して実現していました。

アップダウンコントロールの場合はプロパティに相手のコントロール名と操作対象のプロパティを設定するだけで、自動的に連動機能が確立するようになっています。連動のための特別のコードを書く必要はありません。

最初に作成したテキストボックスコントロールとアップダウンコントロールの組み合わせは、アップダウンコントロールのボタンを操作するとテキストボックスにその値が数値で表示されます。たとえばこれまでは、

```
TextBox1.Text = UpDown1.Value
```

などというコードを作成してこのような動作を行ってきました。しかし、アップダウンコントロールとそれに連結されたコントロールではこのコードがなくても同じ動作を行います。このように、アップダウンコントロールの連結機能を使えば、プログラムの開発が軽減されます。

2 アップダウンコントロールを他のコントロールと連動させる場合は、まずはじめにプロパティページの［連動コントロール］欄か［AutoBuddy］プロパティに、相手のコントロール名を設定します。次に、プロパティページの［連動プロパティ］欄または［BuddyProperty］プロパティで、連動させるプロパティを指定します。

これで、アップダウンコントロールの［Value］プロパティの値が、指定したプロパティの値と連動するようになります。アップダウンコントロールの可動範囲は、［Max］と［Min］プロパティで上限と下限を設定します。また、［Increment］プロパティを使うと、アップダウンコントロールの1回の動作で増減する量を設定でき

ます。アップダウンコントロールの動作量は、［Value］プロパティに格納されます。これらのプロパティは、［プロパティページ］ダイアログボックスの［スクロール］タブでも設定できます。

可動範囲を設定する［プロパティページ］ダイアログボックス

また、［Orientation］プロパティを使用すると、アップダウンコントロールの水平・垂直方向を設定できます。いずれも用意されている定数を設定します。

3　最初に作成したコントロールの組み合わせは、連動コントロールにテキストボックスコントロールを、連動プロパティに［Text］プロパティを指定しています。アップダウンコントロールのボタンをクリックするとテキストボックスにその値が表示されます。

2番目に作成したコントロールの組み合わせは、連動コントロールにラベルコントロールを、連動プロパティに［Width］プロパティを設定しています。そのため、アップダウンコントロールを操作すると、ラベルコントロールの幅が変化します。このような場合は、デザイン時のラベルコントロールの幅と、アップダウンコントロールの［Min］プロパティの値をそろえておくのがポイントです。

3番目に作成したコントロールの組み合わせは、連動コントロールにシェイプコントロールを、連動プロパティに［FillColor］プロパティを設定しています。このシェイプコントロールだけは、アップダウンコントロールの連動コントロール機能の「自動設定」が使えませんでしたが、コントロール名を直接入力することで、連動させています。

4 アップダウンコントロールの［Value］プロパティの中身を表示するために、ラベルコントロールを組み合わせて、コードで［Value］プロパティの値を表示しています。

```
Private Sub UpDown3_Change()
    Label2.Caption = UpDown3.Value
End Sub
```

アップダウンコントロールのボタンがクリックされると、コントロールには「Change」イベントが発生します。クリックした時に何か処理を行わせたい場合は、このイベントプロシージャに処理を記述します。ここでは、アップダウンコントロールの［Value］プロパティの値を、ラベルコントロールの［Caption］プロパティに出力しています。

また、［増加］ボタンまたは［減少］ボタンをクリックしたときに、別々の処理を行わせたい場合は、それぞれ「UpClick」「DownClick」イベントを使用します。

まとめ

　アップダウンコントロールの連動機能は、プログラムのデータ入力処理を組みたてる上で、とても有効な機能を提供してくれます。
　しかし、同期する連動プロパティへの値の代入は、アップダウンコントロールの［Value］プロパティの値が直接代入されるため、式などと組み合わせて代入したい場合は、従来どおりイベントプロシージャと式をコードで記述する必要があります。

練習問題

Q1
　アップダウンコントロールをツールバーコントロールに組み合わせて、アップダウンコントロールのボタンをクリックするとツールバーコントロールのボタンが増減するプログラムを作成してください。

【ヒント】
　連動コントロールの機能は使わずに、アップダウンコントロールの「UpClick」および「DownClick」イベントを使用します。

解答は巻末に→

COLUMN ……

Dimステートメントの記述場所

変数を宣言するDimステートメントですが、本書の各レッスンでは基本的にプロシージャの先頭にまとめて宣言しています。

これは、作成したプログラムコードを後で読み返したり、他の人が読む場合などに、このプロシージャではこれだけの変数を使っている、というのが一目でわかるようにするために、プロシージャの先頭で宣言しているためで、Dimステートメントは必ずプロシージャの先頭でなければ記述できない、というものではありません。

実は、Dimステートメントは、プロシージャのどこにでも記述できます。例えば、Lesson17では、コードの途中でDimステートメントを記述し、宣言したオブジェクト変数を使用しています。

Let's VBSchool!!

第４日 (Lesson12～Lesson14)
時計の作成

Visual Basicの時刻を扱う関数を使用して、他のコントロールなどと組み合わせると、簡単な時計のプログラムが作成できます。

Time関数は、現在の時刻を取得する関数です。この関数をタイマーコントロールのイベントで実行すると、秒刻みで時刻を表示することができます。また、イメージコントロールやプログレスバーコントロールなどと組み合わせて、アニメーションのある時計を作成したり、フォームのキャプションに時刻を表示することが可能になります。

今日のレッスンでは、時刻を扱う関数とコントロールを組み合わせて、いろいろな時計を作成します。

今日のレッスンで解説する項目

- ❑ Time関数の使用方法
- ❑ タイマーコントロールを使った時刻表示
- ❑ プログレスバーの秒表示設定
- ❑ イメージコントロールを使ったアニメーション
- ❑ Shapeコントロールを使ったアニメーション
- ❑ コモンダイアログとラベルコントロールの連動操作
- ❑ 「フォント」ダイアログボックスの設定
- ❑ Format関数による表示形態の設定
- ❑ 時報の組み込みとBeepステートメント
- ❑ フォームのキャプションを時刻表示に使う
- ❑ WindowStateプロパティの操作
- ❑ 複数のタイマーコントロールの操作

Lesson 12

第４日
１時限目

時計の作成

Lesson12.vbp
VB6CD2
Lesson12

アニメーションクロックの作成

時刻を取得する関数「Time」と、タイマーコントロールやラベルコントロール、プログレスバーコントロールを組み合わせて、簡単なアニメーションを実行しながら時刻を表示する時計を作成します。

今回作成するプログラム

アニメーションクロック

20:55:53

指定した時間ごとにイメージが切り替わる

指定した書式で時刻が表示される

時間の進行が黒四角で示される

POINT

Time関数は、Windowsから現在の時刻を取得する関数です。この関数をラベルコントロールなどと組み合わせて、現在の時刻をフォームに表示します。また、タイマーコントロールのTimerイベントでこのTime関数を実行することで、[Interval] プロパティで設定した時間ごとに最新の時刻を表示します。

コントロールとプロパティの設定

※新しいプロジェクトファイル（標準EXE）を用意します。

1 ツールパレットに「Microsoft Windows Common Comtrols 6.0」コンポーネントを追加する

ツールパレットにコモンコントロールが追加される

2 フォームの［Caption］プロパティを「アニメーションクロック」に変更する

プロパティ - Form1	
Form1 Form	
(ｵﾌﾞｼﾞｪｸﾄ名)	Form1
Appearance	1 - 3D
AutoRedraw	False
BackColor	&H8000000F&
BorderStyle	2 - 可変
Caption	アニメーションクロック
ClipControls	True
ControlBox	True

1 「アニメーションクロック」と入力

3 ツールパレットの［Timer］を使って、フォーム上にタイマーコントロールを配置する

アニメーションクロック

1 斜めにドラッグして配置

コンポーネントを追加する

コンポーネントを追加する方法については、「Lesson5 ツールバーコントロールの組み込み」を参照してください。

4 [Interval] プロパティを「1000」(1秒)に設定する

プロパティ - Timer1	
Timer1 Timer	
(オブジェクト名)	Timer1
Enabled	True
Index	
Interval	1000
Left	3765
Tag	
Top	285

1 「1000」に設定

5 ツールパレットの [ImageList] を使って、イメージリストコントロールを配置する

アニメーションクロック

1 斜めにドラッグして配置

6 [プロパティページ] ダイアログボックスを表示する

プロパティ ページ

全般 | イメージ | 色

- 16 x 16(1)
- 32 x 32(3)
- 48 x 48(4)
- ユーザー設定(C)
- MaskColor の使用(U)

高さ(H): 0
幅(W): 0

OK キャンセル 適用(A) ヘルプ

[プロパティページ] ダイアログ

[プロパティページ] ダイアログボックスを表示するには、イメージリストコントロールが選択された状態で、プロパティウィンドウの [(プロパティ)] をクリックし、その右欄にある [...] ボタンをクリックします。

7 ［イメージ］タブをクリックし、［ピクチャの挿入］を使って、次の５つのアイコンを追加する

244.ico ファイル
245.ico ファイル
250.ico ファイル
247.ico ファイル
246.ico ファイル

8 ツールパレットの［Image］を使って、イメージコントロールを配置する

1 斜めにドラッグして配置

9 ツールパレットの［Label］を使って、ラベルコントロールを配置し、次のプロパティを設定する

プロパティ	値
BorderStyle	1 −実線
Caption	(空白)
Alignment	2 −中央揃え

1 斜めにドラッグして配置

イメージの設定

［ピクチャの挿入］ボタンをクリックすると［図の選択］ダイアログボックスが表示されるので、［ファイルの場所］ボックスで「material」フォルダを選び、アイコンファイルを選んで、［開く］ボタンをクリックします。

［キー］には、今回は何も設定しません。

10 ツールパレットの［ProgressBar］を使って、プログレスバーコントロールを配置する

アニメーションクロック

1 斜めにドラッグして配置

11 ■が9個になるように、サイズ変更ハンドルをドラッグして幅を調節する

アニメーションクロック

1 ここをドラッグ

12 ［Max］プロパティを「10」に設定する

ﾌﾟﾛﾊﾟﾃｨ - ProgressBar1

ProgressBar1 ProgressBar

全体 | 項目別

Height	450
Index	
Left	930
Max	10
Min	0
MouseIcon	(なし)
MousePointer	0 - ccDefault

1 「10」に設定

コードの記述

Form1:(General)-(Declarations)

Lesson12 - Form1 (ｺｰﾄﾞ)

(General) | (Declarations)

```
Dim i As Integer, j As Integer
```

Form1:Form-Load

Lesson12 - Form1 (ｺｰﾄﾞ)

Form | Load

```
Private Sub Form_Load()
    i = 1
    j = 1
End Sub
```

Form1:Timer1-Timer

Lesson12 - Form1 (ｺｰﾄﾞ)

Timer1 | Timer

```
Private Sub Timer1_Timer()
    Dim NowTime As String
    Dim NowSec As String

    NowTime = Time
    NowSec = Right(NowTime, 1)
    j = CInt(NowSec)

    NowTime = Format(Time, "hh:mm:ss")
    Label1.Caption = NowTime

    Image1.Picture = ImageList1.ListImages(i).Picture
    i = i + 1
    If i = 6 Then
        i = 1
    End If

    ProgressBar1.Value = j
End Sub
```

解説

1 このプログラムは、次のように動作します。

①1秒ごとにTime関数を実行し、現在の時刻をラベルコントロールに出力して、フォームに表示します。

②同時に、イメージリストコントロールからアイコンイメージを取り出し、フォームに配置したイメージコントロールに、1秒ごとにアイコンを切り替えて表示します。

③プログレスバーコントロールを使って、秒の推移を表示します。これは、10秒単位で繰り返し表示します。このとき、最新の秒の数字を把握し、そこからプログレスバーの表示がスタートするように、秒とプログレスバーの表示をシンクロさせます。

2 Time関数は、実行すると最新の現在時刻をWindowsのシステムから取得し、戻り値として返してきます。この関数を、タイマーコントロールのTimerイベントプロシージャで実行すると、[Interval] プロパティで設定した時間ごとに最新の時刻を取得できます。

すなわち、[Interval] プロパティを「1000」(1秒) に設定して、Time関数を実行すると、1秒ごとの時刻を取得できることになります。

このプログラムでは、このテクニックを使用して、秒単位までの時刻を表示するデジタル時計を作成します。

3 最初に、フォームのコードモジュールの、変数宣言セクションに2つの変数を整数型で宣言します。これらの変数は、あとでイメージをアニメーションさせたり、プログレスバーの操作に使用します。

```
Dim i As Integer, j As Integer
```

変数を宣言したら、フォームのLoadイベントプロシージャで、初期化しておきます。

```
Private Sub Form_Load()
    i = 1
    j = 1
End Sub
```

4 時計とアニメーション、プログレスバーの処理は、すべてタイマーコントロールのTimerイベントプロシージャで行います。
まず最初に、2つの変数を宣言します。この変数は、現在の時刻と秒を格納する変数で、文字列型で宣言します。

```
Private Sub Timer1_Timer()
   Dim NowTime As String
   Dim NowSec As String
```

宣言したら、Time関数を実行して、戻り値を変数「NowTime」に格納します（Time関数に引数はありません）。また、戻り値はVariant型で返ってきますが、それを文字列型の変数に格納します。これで、現在の時刻が文字列として取得できました。

```
NowTime = Time
```

変数に格納したら、Right関数で文字列の一番右の文字を取り出し、変数「NowSec」に格納します。すなわち、秒の1桁部分を取り出します。

```
NowSec = Right(NowTime, 1)
```

取り出したら、CInt関数で文字列から整数型に変換し、変数「j」に格納します。「CInt」関数は、引数に与えた文字列または数式を、整数に変換して返してくるデータ変換関数です（P.30参照）。

これで、現在時刻の秒の1桁の数字が整数に変換されて、変数に格納されます。

```
j = CInt(NowSec)
```

5 今度は、再度Time関数を実行して、現在の時刻を取得します。そして、Format関数を使用して、この時刻を「"hh:mm:ss"」形式に変換して、変数「NowTime」に再度代入しています。

Time関数で取得できる時刻の表示形式は、Windowsのコントロールパネルにある［地域］アプレットの［時刻］タブで設定された表示形式になります。この設定を、プログラムの中で変更したい場合は、Format関数を使用します。この関数は、指定したデータを、指定した書式に変換して返してくる関数です。

この表示形式をFormat関数
変更する

これを使って、時刻の表示形式を変更して、ラベルコントロールの［Caption］プロパティに出力し、フォームに表示します。

```
NowTime = Format(Time, "hh:mm:ss")
Label1.Caption = NowTime
```

この処理は、タイマーコントロールのTimerイベントプロシージャに記述しています。また、タイマーコントロールのIntervalプロパティを「1000」に設定しています。したがって、1秒ごとにこのプロシージャが実行され、時刻の取得が行われてフォームに表示されるようになります。

ワンポイントアドバイス

筆者のコンピュータのコントロールパネルでは、時刻の設定は「午後 10:47」という形式で設定しています。

このプログラムでは、「10:47:18」という秒表示までできる形式に変更して使用します。コンピュータの設定はそのままで、プログラムの中でだけで書式を設定したい場合は、このようにFormat関数を使用して一時的に書式を変更して使用するといいでしょう。

6 ラベルコントロールへの時刻の表示ができたら、今度はイメージコントロールを使った、イメージのアニメーション表示を作成します。

まず、イメージコントロールの［Picture］プロパティに、表示するイメージを設定します。表示するイメージは、イメージリストコントロールの「ListImages」コレクションの［Picture］プロパティを使用して取得します。

イメージリストコントロールに追加したイメージは、ListImageオブジェクトとして扱われます。イメージリストコントロールは、複数のイメージを格納しますから、「ListImageコレクションオブジェクト」となります。

イメージリストコントロールに格納されているイメージにアクセスするには、［ListImages()］プロパティを使用します。()の中に、プロパティページの［インデックス］と同じ番号を指定すれば、そのイメージをオブジェクトとして参照できるようになります。

プロパティページ

たとえば、「日の丸」のイメージにアクセスしたい場合は、このイメージのインデックスが「2」なので、オブジェクトの指定は、次のような記述になります。

```
ImageList1.ListImages(2)
```

このListImageオブジェクトの［Picture］プロパティに、実際のイメージが格納されていますので、これをイメージコントロールの［Picture］プロパティに代入すれば、そのイメージが表示されます。

```
Image1.Picture = ImageList1.ListImages(i).Picture
```

［ListImages()］プロパティのインデックスに変数「i」を使用して、1回Timerイベントが発生するたびに、このListImageオブジェクトのインデックスを変更することで、イメージコントロールの表示イメージを変化させています。

```
i = i + 1
```

また、イメージリストコントロールには、5つまでしかアイコンイメージを格納していませんから、変数「i」が6になったら、iの値を1に戻しています。これで、5つのアイコンイメージが連続で表示されるようになります。

```
If i = 6 Then
    i = 1
End If
```

アニメーションクロック

20:54:22

アイコンが変化する

7 最後に、プログレスバーコントロールの表示を設定します。

プログレスバーコントロールは、［Value］プロパティに数字を与えると、その数字に見合った■を、［Max］と［Min］プロパティの量の比率で表示します。

すでに、変数「j」には、Time関数で取得した秒の1桁が、整数に変換されて格納されています。この値を、プログレスバーコントロールの［Value］プロパティに与えてあげれば、現在の秒の1桁分が■で表示されるようになります。

ここでは、秒の1桁目が「0」であれば、プログレスバーには■は表示されず、「9」で、バーいっぱいに■が表示されるようにしています。

```
ProgressBar1.Value = j
```

以上で完成です。プログラムを動作させると、時刻表示とともに、アイコンイメージとプログレスバーが表示されます。

アニメーションクロック

20:54:40

秒の1桁が「0」だと、プログレスバーは空白

アニメーションクロック

20:55:19

「9」だと、いっぱいに■で埋まる

まとめ

タイマーコントロールのTimerイベントプロシージャ内でTime関数を使用すると、時刻表示が変化する時計のプログラムが作成できます。この時、タイマーコントロールの［Interval］プロパティを1秒に設定しておくのがポイントです。また、イメージコントロールのイメージ表示を行うには、［Picture］プロパティに表示するイメージを設定しますが、ファイルからイメージを設定する場合はLoadPicture関数を使用していました。

しかし、イメージリストコントロールに格納されているイメージを使用する場合は、ListImageオブジェクトの［Picture］プロパティの値をそのまま代入するだけです。

1つのプログラムで複数のイメージを使用する場合は、イメージリストコントロールを使用したほうが、イメージの管理や操作が楽になります。

練習問題

Q1

作成したプログラムでは、タイマーコントロールのTimerイベントプロシージャで時刻表示の処理を行っているため、プログラムが起動した1秒後から、この処理が実行されます。このため、プログラム起動直後は1秒だけ時刻表示がされません。

そこで、プログラム起動時から時刻表示が行われるように、プログラムを修正してください。

アニメーションクロック

起動直後はなにも表示されない

解答は巻末に→

ワンポイントアドバイス

Timerステートメントを使用すると、Windowsの現在時刻を設定できます。

また、Date関数を使用すると、現在の日付を取得でき、Dateステートメントを使用すれば、現在の日付を設定できます。

Lesson 13

第４日
２時限目

時計の作成

Lesson13.vbp
VB6CD2
Lesson13

デジタルクロックの作成

前のレッスンで、Time関数による時刻の取得と活用方法がわかりましたので、今度はグラフィックなどと組み合わせて、少しデザインを加えた時計を作成しましょう。流行のデジタルウォッチをまねした、「Baby-V」という時計を作成します。

今回作成するプログラム

Baby-V

21:16:14

時刻に合わせてグラフィックの表示が変わる

POINT

Shapeコントロールとラベルコントロールを組み合わせて、時刻表示に加えて、秒を図形の点滅で知らせるようにします。また、メニューコントロールとコモンダイアログコントロールを使って、ラベルコントロールの時刻表示のフォントを、自由に変更できるようにします。

コントロールとプロパティの設定

※新しいプロジェクトファイル（標準EXE）を用意します。

1 ツールパレットに「Microsoft Common Dialog Control 6.0」コンポーネントを追加する

ツールパレットにコモンコントロールが追加される

> **コンポーネントを追加する**
> コンポーネントを追加する方法については、「Lesson5 ツールバーコントロールの組み込み」を参照してください。

2 フォームの［Caption］プロパティを「Baby-V」に変更する

プロパティ - Form1

Form1 Form

全体 | 項目別

(オブジェクト名)	Form1
Appearance	1 - 3D
AutoRedraw	False
BackColor	&H00C0FFFF&
BorderStyle	2 - 可変
Caption	Baby-V
ClipControls	True
ControlBox	True
DrawMode	13 - Copy Pen
DrawStyle	0 - 実線 (中心線)

1 「Baby-V」に変更

3 ［BackColor］プロパティを「&H00C0FFFF&」に変更する

プロパティ - Form1

Form1 Form

全体 | 項目別

(オブジェクト名)	Form1
Appearance	1 - 3D
AutoRedraw	False
BackColor	&H00C0FFFF&

パレット | システム

1 ここをクリック

2 ここをクリック

3 ここをクリック

4 ここをクリック

4 ツールパレットの［Shape］を使って、シェイプコントロールを配置する

5 ［Shape］プロパティを「3 －円」に変更する

6 ［FillColor］と［FillStyle］プロパティを次のように設定して水色に塗りつぶす

プロパティ	値
FillColor	&HOOFFFFCO&
FillStyle	0 －塗りつぶし

7 シェイプコントロールをもう１つ配置する

Baby-V

1 斜めにドラッグして配置

8 ツールバーの［コピー］ボタンをクリックし、続けて［貼り付け］ボタンをクリックする

1 ここをクリック

2 ここをクリック

9 [はい] をクリックしてコントロール配列にする

Microsoft Visual Basic

既に同じ名前のコントロール 'Shape2' があります。コントロール配列にしますか?

はい(Y)　いいえ(N)　ヘルプ(H)

1 ここをクリック

10 [貼り付け] ボタンをクリックして、シェイプコントロールのコピーを３つ配置する

Baby-V

貼り付けて配置する

11 最初の四角を [FillColor] と [FillStyle] プロパティを使って「青色」で塗りつぶす

プロパティ	値
FillColor	&H00FF0000&
FillStyle	0 ー塗りつぶし

Baby-V

青色に変更される

12 残り３つの四角を同様の操作で次のプロパティの値を設定し、白色で塗りつぶす

プロパティ	値
FillColor	&H00FFFFFF&
FillStyle	0 ー塗りつぶし

13 ツールパレットの［Label］A を使って、ラベルコントロールを円の中に配置する

1 斜めにドラッグして配置

14 配置したラベルに、次のプロパティを設定する

プロパティ	値
Alignment	2 ー中央揃え
AutoSize	True
BorderStyle	1 ー実線
Caption	（なし）

３つの四角をまとめて選択する

3つの四角をまとめて選択し、一度にプロパティを設定することができます。キーボードのCtrlキーを押しながらコントロールをクリックすると、複数のコントロールをまとめて選択できます。

15 ツールパレットの［Timer］ を使って、タイマーコントロールを配置する

16 ［Interval］プロパティを「1000」に設定する

プロパティ - Timer1	
Timer1 Timer	
全体	項目別
(オブジェクト名)	Timer1
Enabled	True
Index	
Interval	1000
Left	1935
Tag	
Top	120

17 ツールパレットの［CommonDialog］ を使って、コモンダイアログコントロールを配置する

18 ［プロパティページ］の［フォント］タブで、［フォント名］を「MS Pゴシック」に設定する

1 ここをクリック
2 「MS Pゴシック」を設定
3 ここをクリック

［プロパティページ］ダイアログ

［プロパティページ］ダイアログボックスを表示するには、コモンダイアログコントロールが選択された状態で、プロパティウィンドウの［(プロパティ)］をクリックし、その右欄にある［...］ボタンをクリックします。

19 フォームのサイズ変更ハンドルをドラッグし、サイズを調整する

1 ここをドラッグ

20 フォームに対して次のプロパティを設定し、フォームに表示されているボタンを削除する

プロパティ	値
MaxButton	False
MinButton	False

［最小化］ボタンと［最大化］／［元に戻す］ボタンが削除される

21 メニューエディタを起動し、次のメニューを設定する

キャプション	名前	表示
設定	IDM_SETUP	チェックを外す
...フォント	IDM_FONT	チェックのまま

メニュー エディタ
キャプション(P): 設定
名前(M): IDM_SETUP
インデックス(X):
ショートカット(S): (なし)
ヘルプ コンテキスト番号(H): 0
ネゴシエート位置(O): 0 - なし
チェック(C) 有効(E) 表示(V) ウィンドウ リスト(W)
OK キャンセル
次へ(N) 挿入(I) 削除(T)
設定
····フォント

22 完成したフォーム

Baby-V
22:05:03

コードの記述

Form1:(General)-(Declarations)

Lesson13 - Form1 (コード)

(General) | (Declarations)

```
Dim i As Integer
```

Form1:Form-Load

Lesson13 - Form1 (コード)

Form | Load

```
Private Sub Form_Load()
    i = 0
    Label1.Caption = Format(Time, "hh:mm:ss")
End Sub
```

Form1:Form-MouseDown

Lesson13 - Form1 (コード)

Form | MouseDown

```
Private Sub Form_MouseDown(Button As Integer, _
                      Shift As Integer, X As Single, Y As Single)
    If Button = vbRightButton Then
        PopupMenu IDM_SETUP
    End If
End Sub
```

Form1:IDM_FONT-Click

Lesson13 - Form1 (コード)

IDM_FONT | Click

```
Private Sub IDM_FONT_Click()
    With CommonDialog1
        .Flags = cdlCFBoth
        .ShowFont
        Label1.Font.Name = .FontName
        Label1.Font.Size = .FontSize
    End With
End Sub
```

Form1:Timer1-Timer

Lesson13 - Form1 (コード)

Timer1 | Timer

```
Private Sub Timer1_Timer()

    i = i + 1
    If i = 4 Then
        i = 0
    End If
    Shape2(i).FillColor = vbBlue
    If i = 0 Then
        Shape2(3).FillColor = vbWhite
    Else
        Shape2(i - 1).FillColor = vbWhite
    End If

    Label1.Caption = Format(Time, "hh:mm:ss")
End Sub
```

解説

1 このプログラムは、次のように動作します。

①ラベルコントロールに、１秒ごとに変化する時刻を表示します。
②四角いシェイプコントロールは、１秒ごとに青色の点滅を行います。

③フォーム上をマウスの右ボタンでクリックすると、［フォント］メニューがポップアップします。

Baby-V

フォント

22:07:00

右クリックするとメニューがポップアップする

④［フォント］メニューを選ぶと、［フォント］ダイアログボックスが表示され、ラベルコントロールのフォントを自由に選択することができます。ラベルコントロールは、［AutoSize］プロパティを「True」に設定しているので、選択したフォントやサイズによって、コントロールのサイズが変化します。

Baby-V

22:14:02

フォントとサイズが変更されたラベルコントロール

2　では、順番にコードを説明していきましょう。まず最初に、変数の宣言です。フォームの宣言セクションに、変数「i」を整数型で宣言します。この変数は、シェイプコントロールのアニメーションに使用します。

```
Dim i As Integer
```

宣言したら、フォームのLoadイベントプロシージャで、「0」に初期化します。また、プログラム起動時から時刻を表示できるように、Time関数を使って時刻をラベルコントロールの［Caption］プロパティに設定します。このとき、前のレッスンで行ったように、Format関数で時刻表示の形式を設定します。

```
Private Sub Form_Load()
    i = 0
    Label1.Caption = Format(Time, "hh:mm:ss")
End Sub
```

3 フォームの上でマウスの右ボタンが押されると、メニューが表示されるように、PopupMenuメソッドを実行します。

表示するのは［フォント］メニューです。［設定］メニュー自身は非表示に設定していますから、フォームにはメニューは表示されません。したがってこのPopupMenuメソッドでは、直接［フォント］メニューを指定します。

押されたボタンが右ボタンかどうかは、このプロシージャの引数「Button」の値を調べ、Ifステートメントで処理を決定しています。

```
Private Sub Form_MouseDown(Button As Integer, _
                Shift As Integer, X As Single, Y As Single)
    If Button = vbRightButton Then
        PopupMenu IDM_SETUP
    End If
End Sub
```

4 ［フォント］メニューがクリックされると、「IDM_FONT_Click()」イベントプロシージャが呼び出されます。このプロシージャで、コモンダイアログボックスの［フォント］ダイアログボックスを表示し、ユーザーが選択したフォントの種類とサイズを、ラベルコントロールのそれぞれのプロパティに代入しています。

コモンダイアログコントロールで、［フォント］ダイアログボックスを表示するには、「ShowFont」メソッドを実行します。ただし、このメソッドを実行する前に、コモンダイアログコントロールの［Flags］プロパティに、定数「cdlCFBoth」を設定する必要があります。

［Flags］プロパティは、［フォント］ダイアログボックスを使用する場合は、ダイアログボックスに表示するフォントの種類を指定する機能として働きます。このプロパティに何の値も設定しないと、［フォント］ダイアログボックスのフォントリストには、フォントは一切表示されません。定数「cdlCFBoth」は、コンピュータで利用可能なプリンタフォントとスクリーンフォントをダイアログボックスに表示するオプション設定になります。

```
Private Sub IDM_FONT_Click()
    With CommonDialog1
        .Flags = cdlCFBoth
        .ShowFont
```

フォント

フォントがインストールされていません。
コントロール パネルの［フォント］フォルダを開いて、フォントをインストールしてく

OK

［Flags］プロパティを設定しないとエラーが表示される

ワンポイントアドバイス

コモンダイアログコントロールのプロパティシートで、[フォント名] のところに設定したフォントが、ダイアログボックスの [フォント名] に記述され、リストでは選択状態になっています。このフォント名を設定しないと、初期の選択フォントは空欄のままになります。

設定したフォント名がリストに組み込まれている

フォント名を設定しないと初期の選択フォントは空欄のまま

5 [フォント] ダイアログボックスが表示されると、ユーザーはこのダイアログボックスでフォント名やサイズを選択できるようになります。そして、ユーザーがフォント名を選択すると、コモンダイアログコントロールの [FontName] プロパティに、フォントサイズは [FontSize] プロパティに格納されます。

これを、そのままラベルコントロールのFontオブジェクトの、[Name] と [Size] プロパティに代入すれば、ラベルコントロールで表示する文字列のフォントを変更できます。

```
Label1.Font.Name = .FontName
Label1.Font.Size = .FontSize
```

6 最後に、時刻表示の処理を作成します。これは、タイマーコントロールのTimerイベントプロシージャに作成します。

最初に、シェイプコントロールのアニメーション表示を作成します。このコントロールは、1秒ごとに順番に色を白色から青色に点滅させて、点滅表示が回転するように動作させます。四角いシェイプコントロールは、コントロール配列にしてありますので、配列のインデックスに変数を使えば、式でコントロールにアクセスすることができます。これを使って、4つの四角いシェイプコントロールの [FillColor] プロパティを操作し、順番に点滅するように見せかけます。

最初の1秒で青と白の塗りつぶしを入れ替える

まず、変数「i」で塗りつぶすシェイプコントロールの順番を指定します。この変数「i」の値に1を加えておきます。

フォームに配置したシェイプコントロールのオブジェクト名は、「Shape2(0)」から「Shape2(3)」までの4つです。もし、「i」の値が「4」になってしまうと、「Shape2(4)」にアクセスすることになりますが、このコントロールは存在しません。そこで、変数「i」の値を「0」に戻しておきます。

```
Private Sub Timer1_Timer()
    i = i + 1
    If i = 4 Then
        i = 0
    End If
```

7 そして、この変数を使って、シェイプコントロールの［FillColor］プロパティの値を「vbBlue」（青色）に設定します。

```
Shape2(i).FillColor = vbBlue
```

また、青色に塗りつぶしたシェイプコントロールが、最初のコントロール「Shape2(0)」であれば（すなわち変数「i」の値が「0」であれば）、4つめのシェイプコントロールを白色に塗りつぶします。

```
If i = 0 Then
    Shape2(3).FillColor = vbWhite
```

変数「i」の値が「0」以外であれば、1つ前のインデックス番号のシェイプコントロールを白色に塗りつぶします。

```
Else
    Shape2(i - 1).FillColor = vbWhite
End If
```

! ワンポイントアドバイス

「vbBlue」「vbWhite」は、色番号を表す「カラー定数」です。

これで、1秒ごとに、それまで青色だったシェイプコントロールは白色になり、次のシェイプコントロールが青色に塗られます。この動作が1秒ごとに行われるため、四角いシェイプコントロールが点滅を繰り返すように見えます。

8 最後に、ラベルコントロールの［Caption］プロパティに、Format関数を使って、現在の時刻を表示します。

```
Label1.Caption = Format(Time, "hh:mm:ss")
```

これで、プログラムが完成です。

Baby-V

22:17:53

まとめ

シェイプコントロールのように、コントロール配列を使用すると、同じオブジェクト名にインデックス番号を使って、コントロールにアクセスできるようになります。インデックスが使えるということは、式を使うことができますので、自動処理がやりやすくなります。

また、コモンダイアログボックスコントロールの［フォント］ダイアログボックスを使用するときは、表示するフォントの種類を設定するために、［Flags］プロパティに値を設定する必要があることを忘れないでください。

このレッスンでは、説明のために簡単なデザインのデジタルクロックを作成しましたが、皆さんはぜひもっと凝ったデザインの、楽しい時計を作成してみてはいかがでしょう。

練習問題

Q1 このクロックに時報を組み込んでください。サウンドは、ビープ音を使用します。

解答は巻末に→

Lesson 14

第４日
３時限目

時計の作成

Lesson14.vbp
VB6CD2
Lesson14

タスクバーでの時刻表示（アイコンの点滅）

時計のプログラムの３番目は、タスクバーに格納して時刻を表示するプログラムです。フォームのキャプションに時刻を表示するようなプログラムを作成すると、プログラムをタスクバーに格納した時点で、タスクバー上で時刻表示が行えます。この特徴を生かして、普段はタスクバー上で時刻を表示し、通常のウィンドウ表示になれば、フォーム上に時刻を表示するプログラムを作成します。

今回作成するプログラム

タスクバー表示

通常のウィンドウ表示

POINT

フォームの［Caption］プロパティに、Time 関数を使った時刻を設定すると、フォームのキャプションで時刻表示をします。このキャプションは、プログラムをタスクバー化（最小化）してもそのまま表示されます。また、［WindowState］プロパティを使用すると、プログラム起動時に直接タスクバーに格納されるプログラムを作成できます。

これらの機能を使って、タスクバー上で時刻を表示するデジタルクロックを作成します。

コントロールとプロパティの設定

※新しいプロジェクトファイル（標準EXE）を用意します。

1 ツールパレットの［Timer］を使って、タイマーコントロールを2つ配置する

2 それぞれの［Interval］プロパティを「1000」に設定する

(オブジェクト名)	Timer2
Enabled	True
Index	
Interval	1000
Left	3390
Tag	
Top	720

3 ツールパレットの［Label］を使って、ラベルコントロールを配置する

アイコンのコピー

CD-ROMの「Lesson14」フォルダ内にある「Smiley.ico」「Smiley2.ico」の2つのファイルをCドライブにコピーしておいてください。

4 配置したラベルコントロールに、次のプロパティを設定する

プロパティ	値
Alignment	2－中央揃え
BorderStyle	1－実線
Font	MS P ゴシック 18 ポイント
ForeColor	&H00FFFF00（シアン）

5 サイズ変更ハンドルをドラッグして、フォームのサイズを縮小する

1 ここをドラッグ

6 フォームの［BorderStyle］プロパティを「1－固定（実線）」に設定する

1 ここをクリック

2 ここをクリック

3 ここをクリック

7 フォームの［Icon］プロパティに「246.ico」を設定する

アイコンの読み込み

ファイルの場所(I): material

244.ico 245.ico 246.ico 247.ico 250.ico 251.ico 264.ico 266.ico 48.ico 50.ico 52.ico 54.ico 56.ico 58.ico Atoms2.ico Button13.ico ButtonIf.ico Buttonrg.ico Calc.ico Camera.ico center.ico check.ico Clipbrd2.ico Clock8.ico

ファイル名(N): 264.ico

ファイルの種類(T): アイコン (*.ico;*.cur)

開く(O) キャンセル ヘルプ(H)

1 ここをクリック

2 ここをクリック

8 ［WindowState］プロパティを「1 －最小化」に設定する

プロパティ－Form1

Form1 Form

全体 項目別

Icon	(アイコン)
KeyPreview	False
Left	5460
LinkMode	0 － なし
LinkTopic	Form1
MaxButton	False
MDIChild	False
MinButton	True
MouseIcon	True
MousePointer	False
Moveable	True
NegotiateMenus	True

1 ここをクリック

2 ここをクリック

3 ここをクリック

9 ［MinButton］プロパティを「True」に、［MaxButton］プロパティを「False」に設定する

プロパティ－Form1

Form1 Form

全体 項目別

Icon	(アイコン)
KeyPreview	False
Left	5460
LinkMode	0 － なし
LinkTopic	Form1
MaxButton	False
MDIChild	False
MinButton	True
MouseIcon	True
MousePointer	False
Moveable	True
NegotiateMenus	True

1 「False」を設定

2 「True」を設定

アイコンの設定

［Icon］プロパティをクリックし、その右欄の右端にある［...］ボタンをクリックすると、［アイコンの読み込み］ダイアログボックスが表示されます。

「246.ICO」ファイルは、付録CD-ROMの「material」フォルダに収録されています。

コードの記述

Form1:(General)-(Declarations)

Lesson14 - Form1 (コード)

(General) | (Declarations)

```
Dim Pos As Boolean
```

Form1:Form-Load

Lesson14 - Form1 (コード)

Form | Load

```
Private Sub Form_Load()
    Pos = True
    Form1.Caption = Time
End Sub
```

Form1:Timer1-Timer

Lesson14 - Form1 (コード)

Timer1 | Timer

```
Private Sub Timer1_Timer()
    Dim NowTime As Variant

    NowTime = Time

    If Form1.WindowState = vbNormal Then
        Form1.Caption = "Taskbar Clock"
    Else
        Form1.Caption = NowTime
    End If

    Label1.Caption = NowTime
End Sub
```

Form1:Timer2-Timer

Lesson14 - Form1 (ｺｰﾄﾞ)

Timer2 ▼　Timer ▼

```
Private Sub Timer2_Timer()
    If Pos = True Then
        Form1.Icon = LoadPicture("c:¥SMILEY.ICO")
        Pos = False
    Else
        Form1.Icon = LoadPicture("c:¥SMILEY2.ICO")
        Pos = True
    End If
End Sub
```

解 説

1 このプログラムは、次のように動作します。

①プログラムが起動すると、すぐにWindowsのタスクバーに格納されます。

②この状態で、時刻の表示とアイコンのアニメーションを行います。

③タスクバーにあるプログラムをマウスでクリックすると、プログラムは通常のウィンドウ表示になります。この状態では、フォームのキャプションには、アニメーションしないアイコンとプログラム名が表示され、ラベルコントロールに時刻が表示されます。

プログラムの表示形態が「標準」から「最小化」に変化すると、フォームには「Resize」というイベントが発生します。このイベントをうまく使用すると、フォームの表示状態によってプログラムの動作を変更させることができます。

このプログラムでは、これらのテクニックを使用して、時刻の表示を切り替えています。

また、フォームの［WindowState］プロパティを、デザイン時に「1－最小化」に設定しています。こうしておくと、プログラムが起動するとすぐにタスクバーに格納されてプログラムが実行します。

2 では、コードを作成していきましょう。最初にフォームの宣言セクションに、変数を宣言します。「Pos」はアイコンの表示を切り替えるのに使用します。この変数は、２つの値しか使用しないため、論理型（Boolean）で宣言します。また、複数のプロシージャで使用するため、モジュールレベルの変数として宣言します。

```
Dim Pos As Boolean
```

次に、フォームのLoadイベントプロシージャで、変数「Pos」を初期化します。論理型で宣言しているため、使用できる値は「True」と「False」の2つしかありません。ここでは、「True」に初期化します。

また、プログラム起動時にタスクバーに格納されたらすぐに時刻の表示を行うため、Time関数の戻り値をフォームの［Caption］プロパティに代入しておきます。これで、フォームのキャプション部分に時刻が表示されるようになります。

```
Private Sub Form_Load()
    Pos = True
    Form1.Caption = Time
End Sub
```

3 最初のタイマーコントロールのTimerイベントプロシージャで、時刻の表示処理を行います。まず、Time関数の戻り値を、変数「NowTime」に格納しておきます。Time関数は、時刻をバリアント型で返してきますので、受け取る変数は同じ型で宣言します。

```
Private Sub Timer1_Timer()
    Dim NowTime As Variant

    NowTime = Time
```

次に、Ifステートメントで［WindowState］プロパティの値を調べます。フォームの値を参照すると、現在のフォームの表示状態を把握することができます。たとえば、［WindowState］プロパティに定数「vbMinimized」が格納されていれば、現在フォームは「最小化」（タスクバーに格納）状態になっていることになります。もし、標準のウィンドウ表示状態であれば、フォームの［Caption］プロパティには、プログラムの名前を代入して表示します。

```
If Form1.WindowState = vbNormal Then
    Form1.Caption = "Taskbar Clock"
```

標準のウィンドウ表示でなければ（すなわち最小化状態であれば）、フォームの［Caption］プロパティには、Time関数の戻り値である現在の時刻を代入します。

```
Else
    Form1.Caption = NowTime
End If
```

そして、フォームのラベルコントロールにも、現在時刻を表示しておきます。

```
Label1.Caption = NowTime
```

これは、ユーザーによってプログラムが標準のウィンドウ表示になったときに、すぐに時刻を表示できるようにしておくためです。

4 今度は、もう1つのタイマーコントロール「Timer2」で、フォームのアイコンをアニメーションさせる処理を作成します。この仕組みはいたって簡単で、フォームの［Icon］プロパティに、Cドライブにある「SMILEY.ICO」と「SMILEY2.ICO」の2つのアイコンファイルを交互に表示することで、アニメーションを実現しています。

変数「Pos」でどちらのアイコンが表示されているのかを判断し、もう一方のアイコンに入れ替えるだけです。フォームの［Picture］プロパティにアイコンイメージを組み込むには、LoadPicture関数を使用します。

```
Private Sub Timer2_Timer()
    If Pos = True Then
        Form1.Icon = LoadPicture("C:¥SMILEY.ICO")
        Pos = False
    Else
        Form1.Icon = LoadPicture("C:¥SMILEY2.ICO")
        Pos = True
    End If
End Sub
```

以上で、プログラムは完成です。

5 プログラムを起動すると、すぐにタスクバーに格納されて、時刻表示とアイコンのアニメーションが行われます。また、このプログラムをクリックすると、通常のフォーム表示になり、フォームのキャプションがプログラム名に変わって、ラベルコントロールに時刻が表示されます。フォームのアイコンは、そのままアニメーションを継続します。

また、時刻表示とアイコンのアニメーションは、別々のタイマーコントロールを使用していますから、それぞれの動作時間を変えて使用することもできます。

まとめ

このプログラムでは、フォームの［WindowState］プロパティを使ってフォームの表示状態を把握して、プログラムの動作を切り替えるテクニックを使用しました。
また、フォームの［BorderStyle］プロパティを「1 −固定（実線）」に切り替えると、ユーザーはフォームのサイズを変更できなくなりますので、固定表示のフォームにすることができます。［MaxButton］プロパティを「False」にすれば、フォームの最大化表示もできなくなります。プロパティの目的に合わせて、フォームの表示操作を設定するこれらのプロパティをうまく組み合わせて使用してください。
また、複数のタイマーコントロールを使用し、［Interval］プロパティの値を変えると、それぞれ違った動作のアニメーションを作成できます。

練習問題

Q1 作成したこのプログラムに、マウスの右ボタンでクリックするとポップアップするメニューを組み込み、ラベルコントロールのフォントと色を変更できるようにしてください。

【ヒント】
コモンダイアログコントロールの［フォント］と［色］ダイアログボックスを使いましょう。

解答は巻末に→

ワンポイントアドバイス

フォームの表示形態が変更されると、フォームには「Resize」イベントが発生します。
表示形態の変更に伴う処理を行いたい場合は、このイベントプロシージャを活用してください。

Let's VBSchool!!

第５日 (Lesson15～Lesson17)

マルチドキュメントビューワの作成

Learning School

Visual Basic には、１つのフォームの中に複数のフォームを表示するアプリケーションを作成できます。この機能を、「MDI」(Multi Document Interface) と言います。プログラムに MDI 機能を組み込むと、１つのプログラムでいくつもの文書やグラフィックイメージなどを表示・編集することができます。また、あらかじめ複数のフォームを用意しなくても、プログラム実行中に、好きな数だけフォームを作成することができます。

今日のレッスンでは、最初に MDI 機能の使い方を覚え、以降のレッスンではこの MDI 機能とリッチテキストボックスコントロールやイメージコントロールを使って、いくつもの文書ファイルやイメージファイルを表示することができる、「マルチドキュメントビューワ」というプログラムを作成します。

今日のレッスンで解説する項目

- ❏ MDI フォームモジュールの組み込み
- ❏ フォームの［MDIChild］プロパティとインスタンスの作成
- ❏ Dim...New ステートメントの使用方法
- ❏ メニューコントロールの「ウィンドウリスト」プロパティの機能
- ❏ MDI 機能の特徴
- ❏ インスタンスの元フォームの性質の継承
- ❏ ツールバーコントロールによる子フォームの生成
- ❏ スタートアップモジュールの指定
- ❏ 種類の違う子フォームの作成
- ❏ コモンダイアログボックスの実行時のプロパティ設定
- ❏ ［Function］プロパティの使用方法
- ❏ MDI フォームの［Arrange］プロパティの使用方法
- ❏ フォームとコントロールのサイズを連動させる処理

Lesson 15

第5日
1時限目

マルチドキュメントビューワの作成

Lesson15.vbp
VB6CD2
Lesson15

MDIプログラムの作成

MDI機能を持ったプログラムを作成しながら、MDI機能の実装方法やプログラム実行時の「子フォーム」の生成方法などを身に付けていきます。

今回作成するプログラム

MDIForm1
ファイル　ウィンドウ
Form1　追加
Add Form1
Add Form2
Add Form3
これはForm1です。

[追加] をクリックすると...

新しいフォームが追加される

MDIForm1
ファイル　ウィンドウ
1 Add Form1
2 Add Form2
3 Add Form3
4 Add Form4
5 Add Form5
6 Add Form6
Add Form4
Add Form5
Add Form6
これはForm1です。

[ウィンドウ] メニューでフォーム名をクリックすると...

その名前のフォームが前面に表示される

POINT

親フォームに「MDIフォーム」を使用し、これに配置したツールバーコントロールとメニューコントロールを使って、プログラム内に次々とフォームを生成していきます。

コントロールとプロパティの設定

※新しいプロジェクトファイル（標準EXE）を用意します。

1 ツールパレットに「Microsoft Windows Common Comtrols 6.0」コンポーネントを追加する

ツールパレットにコモンコントロールが追加される

2 ［標準］ツールバーのボタンを使って、プロジェクトに「MDIフォームモジュール」を追加する

1 ここをクリック

2 ここをクリック

3 ここをクリック

3 フォーム「Form1」をクリックし、［MDIChild］プロパティを「True」に設定する

1 ここをクリック

2 ここをクリック

3 ここをクリック

コンポーネントを追加する

コンポーネントを追加する方法については、「Lesson5 ツールバーコントロールの組み込み」を参照してください。

MDIフォーム

［MDIフォームモジュール］をクリックし、表示される［MDIフォームモジュールの追加］ダイアログボックスで［MIDフォームモジュール］が選択されていることを確認して［開く］をクリックすると、「MDIForm1」フォームウィンドウが追加されます。

ラベルコントロールの追加

ラベルコントロールを追加するには、ツールパレットの［Label］A を使います。

4 ラベルコントロールを配置し、［Caption］プロパティを「これはForm1です。」に変更する

Form1

これはForm1です。

配置して［Caption］プロパティを変更したラベルコントロール

5 ［プロジェクト］メニューの［Project1 のプロパティ］を選択する

1 ここをクリック

2 ここをクリック

6 ［全般］タブで、［スタートアップの設定］を［MDIForm1］に変更し、［OK］をクリックする

1 ここをクリック

2 ここをクリック

3 ここをクリック

7 サイズ変更ハンドルをドラッグして、「MDIForm1」のサイズを広げる

1 ここをドラッグ

8 ツールパレットの［Toolbar］ を使って、ツールバーコントロールを「MDIForm1」に配置する

9 ［プロパティページ］の［ボタン］タブで［ボタンの挿入］をクリックし、ボタンを2つ挿入する

インデックス	キャプション
1	Form1
2	追加

1 ここをクリック

2 キャプションを設定

［プロパティページ］ダイアログ

［プロパティページ］ダイアログボックスを表示するには、プロパティウィンドウの［(プロパティページ)］をクリックし、右欄の［...］ボタンをクリックします。

ボタンを追加する方法

ボタンを追加する方法の詳しい説明については、「Lesson 5　ツールバーコントロールの組み込み」を参照してください。

10 「MDIForm1」を選択してメニューエディタを起動し、次のメニューを設定する

キャプション	名前
ファイル	IDM_FILE
....Form1	IDM_OPEN
....フォームの追加	IDM_ADD
....-	IDM_SEPARATOR
....終了	IDM_EXIT
ウィンドウ	IDM_WINDOW

11 「ウィンドウ」を選択し、[ウィンドウリスト] ボックスをチェックする

1 「ウィンドウ」を選択

2 ここをチェック

コードの記述

MDIForm1:(General)-(Declarations)

Lesson15 - MDIForm1 (ｺｰﾄﾞ)
(General) / (Declarations)

```
Dim i As Integer
```

MDIForm1:MDIForm-Load

Lesson15 - MDIForm1 (ｺｰﾄﾞ)
MDIForm / Load

```
Private Sub MDIForm_Load()
    i = 1
End Sub
```

MDIForm1:Toolbar1-ButtonClick

Lesson15 - MDIForm1 (ｺｰﾄﾞ)
Toolbar1 / ButtonClick

```
Private Sub Toolbar1_ButtonClick(ByVal Button As _
    MSComctlLib.Button)
    Select Case Button.Index
        Case 1
            Form1.Show
        Case 2
            Call AddForm
    End Select
End Sub
```

MDIForm1:(General)-AddForm

Lesson15 - MDIForm1 (ｺｰﾄﾞ)
(General) / AddForm

```
Private Sub AddForm()
    Dim AddForm As New Form1

    With AddForm
        .Show
        .Caption = "Add Form" & i
    End With
    i = i + 1
End Sub
```

MDIForm1:IDM_OPEN-Click

Lesson15 - MDIForm1 (コード)

IDM_OPEN | Click

```
Private Sub IDM_OPEN_Click()
    Form1.Show
End Sub
```

MDIForm1:IDM_ADD-Click

Lesson15 - MDIForm1 (コード)

IDM_ADD | Click

```
Private Sub IDM_ADD_Click()
    Call AddForm
End Sub
```

MDIForm1:IDM_EXIT-Click

Lesson15 - MDIForm1 (コード)

IDM_EXIT | Click

```
Private Sub IDM_EXIT_Click()
    Unload Me
    End
End Sub
```

解　説

1　このプログラムは、次のように動作します。

①ツールバーの［Form1］ボタンまたは［ファイル］メニューの［Form1］をクリックすると、標準フォーム「Form1」が表示されます。

MDIForm1
ファイル　ウィンドウ
Form1　追加
Form1
これはForm1です。

ここをクリックすると…

新しい標準フォームが
作成される

②ツールバーの［追加］ボタン、または［ファイル］メニューの［追加］をクリックすると、新しいフォームが追加されます。このフォームの追加は、コンピュータのメモリが許す限り、いくつでも行えます。

③［ウィンドウ］メニューをクリックすると、表示しているフォームの名前がリスト表示されます。

プログラムデザイン時は、フォームは1つしか配置していませんが、ツールバーの［追加］ボタンをクリックすれば、いくつでもフォームが表示できます。

このように、1つのフォームを作っておくだけで、同じ属性を持ついくつものフォームを表示できるのが、MDIプログラムの特徴です。プログラム実行時に、ユーザーが必要に応じて好きなだけフォームを作成して使うことができるのも、MDIプログラムの大きな特徴です。

2　MDIプログラムを作成する場合は、プログラムのベースに通常のフォームではなく「MDIフォーム」という専用のフォームを使用します。このフォームは、Visual Basicのツールバーまたは［プロジェクト］メニューにある［MDIフォームモジュールの追加］をクリックすると、プロジェクトファイルに挿入できます。ただし、MDIフォームは、1つのプロジェクト（プログラム）に1つしか挿入できません。

[MDIフォームモジュールの追加]メニュー

そして、このMDIフォームの中に、いくつもフォームを作成して表示します。通常、MDIフォームを「親フォーム」、MDIフォーム内に表示されるフォームを「子フォーム」と呼びます。

親フォーム内に子フォームが表示される

親フォーム内に表示された子フォームは、親フォーム内であれば、最大化表示や最小化表示が可能になります。最小化は、親フォーム内にアイコン表示されます。

最大化した子フォーム

最小化（アイコン化）した子フォーム

3　このMDIフォームがプログラムのベースになりますので、プログラム起動時にMDIフォームが表示されるように、スタートアップの設定を変更しておく必要があります。そうしないと、標準のフォーム（Form1）が表示されてしまい、MDI機能が使えなくなってしまいます。

スタートアップの設定は、Visual Basicの［プロジェクト］メニューの［Project1のプロパティ］を選択すると表示される［プロジェクトプロパティ］ウィンドウで行います。［スタートアップの設定］リストボックスにある「MDIForm1」を選べば、このフォームをプログラム起動時のフォームに指定できます。

「MDIForm1」を選択

4　プログラム実行時に子フォームを生成する仕組みは、2つあります。

1つは、フォームの［MDIChild］プロパティにあります。まず、子フォームの元になる標準フォームを1つ用意します。そして、フォームの［MDIChild］プロパティを「True」に設定します。これで、このフォームは、MDIフォームの子フォームとして機能するようになります。

もう1つの仕組みは、コードにあります。このフォームを元にコードで「New」キーワードを使って、元のフォームの複製（これを「インスタンス」と呼びます）を作成します。

これで、複数の子フォームを作成できるようになります。ですから、必ずインスタンスの元になるフォームが、プロジェクトに挿入されていなければなりません。

次のコードは、標準フォーム「Form1」を元に、「AddForm」という名前の子フォーム（インスタンス）を生成します。

```
Dim AddForm As New Form1
```

項目	説明
Dim	必ず、Dimステートメントで宣言します。
AddForm	生成されるインスタンスのオブジェクト名を指定します。任意のオブジェクト名を付けられます。
New	「New」キーワードです。Dimステートメントの中で使用すると、新しいオブジェクトを作成します。
Form1	作成するオブジェクトの元を指定します。この場合は、標準フォーム「Form1」のクローンが作成されます。

5 作成された子フォームは、特に手を加えなければ、元になったフォームの性質をそのまま受け継ぎます。たとえば、このプログラムでは、「Form1」に配置したラベルコントロールの文字「これはForm1です。」が、生成された子フォームにも同じように表示されています。

同じメッセージが
表示される

試しに、「Form1」の［BackColor］プロパティを変えて、再度子フォームを生成してみましょう。「Form1」と同じ色の子フォームが作成されます。

同じ色の子フォームが
作成される

6 では、プログラムのコードを作成していきましょう。注意して欲しいのは、MDIフォームのコードは、MDIフォームのコードウィンドウに作成する、ということです。これまでは、標準フォームにコードを作成してきましたが、MDIフォームの操作は、MDIフォームのコードウィンドウに作成します。

まず、MDIフォームをダブルクリックして、コードウィンドウを開いてください。現在、どのモジュールのコードを作成しているのかは、コードウィンドウのキャプションを見ればわかります。

「MDIForm1」のコードウィンドウであることを、オブジェクトリストボックスで確認できる

ワンポイントアドバイス

コードウィンドウは、モジュールごとに用意されています。処理を実行したいモジュールごとに、それぞれのコードウィンドウに処理を記述していきます。また、プロパティウィンドウも、モジュールごと（フォームごと）に、配置したコントロールのプロパティが設定できるようになっています。プロジェクトに複数のフォームモジュールを配置した場合は、どのモジュールのコントロールなのかをきちんと確認してから、プロパティの操作を行ってください。

7 コードウィンドウが表示されたら、まずはじめに宣言セクションに変数「i」を宣言します。

```
Dim i As Integer
```

この変数は、子フォームを作成したときに、何番目の子フォームなのかがわかるように、子フォームのキャプションに番号を付けるために使用します。宣言したら、MDIフォームのLoadイベントプロシージャで、変数「i」を初期化します。くれぐれも、対象のフォームを間違えないでください。このプログラムでは、プログラム起動時はMDIフォームが表示されているようになっています。ですから、プログラム起動時のいろいろな初期化処理は、このMDIFormのLoadイベントプロシージャに作成する必要があります。

```
Private Sub MDIForm_Load()
    i = 1
End Sub
```

8

今度は、ツールバーコントロールのボタンの処理を作成します。すでに説明したように、ツールバーコントロールのいずれかのボタンがクリックされると、このボタンの「ButtonClick」イベントプロシージャが呼び出されます。そして、クリックされたボタンの種類は、Buttonオブジェクトの［Index］または［Key］プロパティを参照することでわかるようになっています。

MDIフォームをデザインしたときは、ツールバーコントロールの各ボタンにはインデックス番号だけしか付けていませんので、［Index］プロパティの値を調べることで、クリックされたボタンを判別します。

クリックされたボタンの種類は、Select Caseステートメントで判断し、「Form1」ボタンがクリックされれば、「Show」メソッドで「Form1」フォームを表示します。［追加］ボタンがクリックされれば、次に作成するForm1のインスタンス（子フォーム）を生成する、ユーザー定義のプロシージャ「AddForm」を呼び出します。

```
Private Sub Toolbar1_ButtonClick(ByVal Button As _
                                MSComctlLib.Button)
    Select Case Button.Index
        Case 1
            Form1.Show
        Case 2
            Call AddForm
    End Select
End Sub
```

9

子フォームを作成する処理は、ユーザー定義のプロシージャ「AddForm」に記述します。わざわざユーザー定義のプロシージャにしたのは、この処理は、あとでメニューコントロールの「追加」メニューでも使用できるようにするためです。

このプロシージャでは、最初にDim...Newステートメントで、「Form1」のインスタンスを「AddForm」という名前で生成します。そして、生成した「AddForm」を「Show」メソッドで表示します。

```
Private Sub AddForm()
    Dim AddForm As New Form1

    With AddForm
        .Show
```

新しいオブジェクト「AddForm」は、標準フォーム「Form1」のクローンですから、「Form1」の持っているすべてのプロパティとメソッド、イベントとまったく同じ物を使用できます。そこで、ここでは［Caption］プロパティに、「Add Form」という文字列と変数「i」を連結して設定します。

```
        .Caption = "Add Form" & i
    End With
    i = i + 1
```

このインスタンスの生成は、ツールバーコントロールの［追加］ボタンがクリックされるたびに、1回ずつ行われます。ボタンを2回クリックすれば、インスタンスは2つ作成されます。「i = i + 1」という式を組み込んでいるために、インスタンスが作成されるごとに、変数「i」の値は1つずつ増加されて、子フォームのキャプションには、「Add Form1」というように、今作成されたインスタンスが何番目に作成されたのかが表示されます。

10

最後に、メニューコントロールの処理を作成します。

「Form1」メニューのClickイベントプロシージャでは、そのままShowメソッドで「Form1」を表示します。［追加］メニューのClickイベントプロシージャでは、子フォームを作成するユーザー定義のプロシージャ「AddForm」を呼び出します。

最後の「終了」メニューのClickイベントプロシージャは、すっかりおなじみになった、プログラムの終了処理です。Unloadステートメントのオブジェクト指定に、「Me」キーワードを使用しています。

```
Private Sub IDM_OPEN_Click()
    Form1.Show
End Sub

Private Sub IDM_ADD_Click()
    Call AddForm
End Sub

Private Sub IDM_EXIT_Click()
    Unload Me
    End
End Sub
```

11 メニューコントロールの［ウィンドウ］メニューは、特に何の処理も組み込んでいません。しかし、メニューエディタでこのメニューだけ［ウィンドウリスト］にチェックを付けています。このオプションがチェックされていると、開いている子フォームの［Caption］プロパティの値が、ウィンドウ名としてメニューに自動的に組み込まれます。そして、このウィンドウ名をクリックすると、そのフォームがアクティブに（最前面に）なります。

COLUMN

インスタンスについて

「インスタンス（instance）」というのは直訳すると「例」、「場合」、「事実」などという意味ですが、わかりにくければ「コピー」という風に考えてもいいでしょう。つまり、プログラムの実行時に生成されるオブジェクト実体のことを指します。

このレッスンの場合、インスタンスが生成されると全く同じ属性を持つオブジェクト（フォーム）のコピーができあがります。このコピーは元のフォームと全く同じコントロールとコードを共有しています。複数のウィンドウを持つアプリケーションを作成するときは、このようなMDI機能をうまくプログラムに組み込むことで、メモリを節約することができます。逆にウィンドウごとにフォームモジュールを用意するようなプログラムでは、メモリをどんどん消費していってしまい、メモリ不足を引き起こすことがあります。注意しましょう。

まとめ

MDI機能を使用するには、必ずMDIフォームモジュールをプロジェクトに組み込み、このモジュールをプログラムのスタートアップモジュールに指定してください。標準フォームをMDIフォームの子フォームにする場合は、そのフォームの「MDIChild」プロパティを「True」に設定します。このプロパティを設定しないと、MDIフォーム内にフォームが表示されません。

また、複数の子フォームをプログラム実行時に作成する場合は、子フォームにしたフォームを元に、「New」キーワードを持ったDimステートメントを使用して、フォームのインスタンスを生成します。生成されたインスタンスは、原則としてインスタンスのもとになったフォームの、すべてのプロパティとメソッド、イベントを受け継ぎます。

練習問題

Q1　インスタンスの元になるフォームにコマンドボタンを配置し、メッセージボックスを表示するようにしてください。

メッセージボックスのメッセージは、コマンドボタンが組み込まれている子フォームのCaptionプロパティの値を表示してください。たとえば、次の画面のように、「Add Form3」に組み込まれているコマンドボタンが押されれば、「このフォームは Add Form3」というメッセージを表示するようにします。

また、フォームに配置したラベルコントロールの文字も、「このフォームは Add Form3」というように、フォームの名前が付いた文字になるように修正してください。

【ヒント】

Meキーワードを使います。

ラベルコントロールも
メッセージボックスも、
そのフォームの名前が
付いている

解答は巻末に→

Lesson 16

第５日
２時限目

マルチドキュメントビューワの作成

Lesson16.vbp
VB6CD2
Lesson16

ユーザーインターフェイスの作成

今度は、このMDI機能を使用したアプリケーションを作成しましょう。リッチテキストコントロールとイメージコントロールを組み合わせて、テキストファイル（またはRTFファイル）、およびビットマップやJPEGイメージファイルをいくつでも表示できる、「マルチドキュメントビューワ」です。最初は、ユーザーインターフェイスの作成です。

今回作成するプログラム

マルチドキュメントビューワ
ファイル　ウィンドウ
Exit　Text　Image
JPG、BMP、GIF形式のグラフィックファイルを開きます

マルチドキュメントビューワのユーザーインターフェイスを作成する

POINT

２つの標準フォームを使い、１つのフォームにはリッチテキストボックスコントロールを、もう１つのフォームにはイメージコントロールを配置することで、テキストファイル（またはRTFファイル）とイメージファイルを、それぞれの子フォームに表示します。このレッスンでは、ユーザーインターフェイスの作成を行います。

コントロールとプロパティの設定

※新しいプロジェクトファイル（標準EXE）を用意します。

1 ツールパレットに、次のコンポーネントを追加する

- Microsoft Common Dialog Control 6.0
- Microsoft Windows Common Controls 6.0
- Microsoft Rich Textbox Control 6.0

ツールパレットにコモンコントロールが追加される

2 ［標準］ツールバーのボタンを使って、プロジェクトに「MDIフォームモジュール」を追加する

1 ここをクリック

2 ここをクリック

コンポーネントを追加する

コンポーネントを追加する方法については、「Lesson5 ツールバーコントロールの組み込み」を参照してください。

MDIフォーム

［MDIフォームモジュール］をクリックし、表示される［MDIフォームモジュールの追加］ダイアログボックスで［MIDフォームモジュール］が選択されていることを確認して［開く］をクリックすると、「MDIForm1」フォームウィンドウが追加されます。

3 MDIフォームのサイズを広げ、［Caption］プロパティを「マルチドキュメントビューワ」に変更する

(オブジェクト名)	MDIForm1
Appearance	1 - 3D
AutoShowChildren	True
BackColor	&H8000000C&
Caption	マルチドキュメントビューワ
Enabled	True
Height	5490
HelpContextID	0
Icon	(ｱｲｺﾝ)
Left	0

1 ここを「マルチドキュメントビューワ」に変更

4 ツールパレットの［Toolbar］ を使って、ツールバーコントロールを配置する

1 斜めにドラッグして配置

5 ツールパレットの［ImageList］ を使って、イメージリストコントロールを配置する

1 斜めにドラッグして配置

6 イメージリストコントロールの［プロパティページ］の［イメージ］タブで、次のピクチャを挿入する

インデックス	ファイル名
1	check.ico
2	Doc05.ico
3	Paint_2.ico

ピクチャの挿入

［イメージ］タブをクリックし、［ピクチャの挿入］ボタンをクリックします。［図の選択］ダイアログボックスが表示されるので、挿入するピクチャを選択し、［開く］ボタンをクリックします。

挿入したボタン

7 ツールバーコントロールの［プロパティページ］で、［イメージリスト］に「ImageList1」を指定する

1 ここをクリック

8 ［ボタン］タブで、［ボタンの挿入］を使ってボタンを３つ挿入し、次のプロパティを設定する

インデックス	キャプション	イメージ
1	Exit	1
2	Text	2
3	Image	3

設定するプロパティ

［プロパティページ］ダイアログ

［プロパティページ］ダイアログボックスを表示するには、コントロールを選択し、プロパティウィンドウの［(プロパティページ)］をクリックし、右欄の［...］ボタンをクリックします。

9 ツールパレットの［CommonDialog］ を使って、コモンダイアログコントロールを配置する

10 ［プロパティページ］の［開く/保存］タブで、次のプロパティを設定する

プロパティ	値
タイトル	ファイルを開く
初期ディレクトリ	c:¥

11 メニューエディタを起動し、次のメニューを設定する

キャプション	名前
ファイル	IDM_FILE
....開く	IDM_OPEN
........文書ファイル	IDM_TEXT
........イメージファイル	IDM_IMAGE
....-	separator
....終了	IDM_EXIT
ウィンドウ	IDM_WINDOW
....水平に並べて表示	IDM_HORI
....縦に並べて表示	IDM_VERT
....重ねて表示	IDM_CASCADE

12 「ウィンドウ」を選択し、［ウィンドウリスト］ボックスをチェックする

メニュー エディタ
キャプション(P): ウィンドウ
名前(M): IDM_WINDOW
インデックス(X):
ショートカット(S): (なし)
ヘルプ コンテキスト番号(H): 0
ネゴシエート位置(O): 0 - なし
チェック(C)　有効(E)　表示(V)　ウィンドウ リスト(W)
OK　キャンセル　次へ(N)　挿入(I)　削除(T)
ファイル
····開く
········文書ファイル
········イメージファイル
····-
····終了
ウィンドウ
····水平に並べて表示
····縦に並べて表示
····重ねて表示

1 「ウィンドウ」を選択

2 ここをチェック

マルチドキュメントビューワ
ファイル　ウィンドウ
開く　文書ファイル　イメージファイル
終了

マルチドキュメントビューワ
ファイル　ウィンドウ
水平に並べて表示
縦に並べて表示
重ねて表示
Exit

作成したメニュー

メニューエディタを起動する

メニューエディタを起動するには、［標準］ツールバーの［メニューエディタ］ボタンをクリックします。

メニューを作成する方法

メニューを作成する詳しい方法については、「Lesson2 合計計算処理の作成」を参照してください。

Projectの名前

[Project1のプロパティ]コマンドの「Project1」の部分には、作業中のプロジェクトを保存したときに付けた名前が表示されます。この例では、「Lesson16」という名前が付けられています。

13 [プロジェクト]メニューの[Project1のプロパティ]を選択する

1 ここをクリック

2 ここをクリック

14 [全般]タブで、[スタートアップの設定]を[MDIForm1]に変更し、[OK]をクリックする

1 ここをクリック

2 ここをクリック

3 ここをクリック

フォームの選択

必要なフォームを選択して前面に表示するには、プロジェクトウィンドウで、必要なフォームの名前をダブルクリックします。

15 Form1を選択し、[MDIChild]プロパティを「True」に設定する

1 ここをクリック

2 ここをクリック

3 ここをクリック

16 ツールパレットの［RichTextBox］を使ってForm1にリッチテキストボックスを配置する

Form1
RichText

1 斜めにドラッグして配置

17 リッチテキストコントロールに、次のプロパティを設定する

プロパティ	値
ScrollBars	3 – rtfBoth
Text	（なし）

18 ［標準］ツールバーのボタンを使って、プロジェクトにフォームモジュール（Form2）を追加する

Lesson16 - Microsoft Visual Basic [デザイン]
ファイル(F) 編集(E) 表示(V) プロジェクト(P) 書式(O) デバッグ(D) 実行(R) クエリー(U)
フォーム モジュール(F)
MDI フォーム モジュール(I)
標準モジュール(M)
クラス モジュール(C)
ユーザー コントロール(U)
プロパティ ページ(R)
ユーザー ドキュメント(D)
DHTML Page
Data Report
WebClass
Data Environment
ファイルの追加(A)... Ctrl+D
- Form1 (Form)

1 ここをクリック
2 ここをクリック

19 [MDIChild] プロパティを「True」に設定する

プロパティ - Form2	
Form2 Form	
LinkTopic	Form2
MaxButton	True
MDIChild	True
MinButton	True / False
MouseIcon	
MousePointer	0 - 既定値
Moveable	True
NegotiateMenus	True

1 ここをクリック
2 ここをクリック
3 「True」に設定

20 ツールパレットの [Image] を使って、「Form2」にイメージコントロールを配置する

Form2

1 斜めにドラッグして配置

21 [BorderStyle] プロパティを「1 ー実線」に設定する

プロパティ - Image1	
Image1 Image	
(オブジェクト名)	Image1
Appearance	1 - 3D
BorderStyle	1 - 実線
DataField	0 - なし
DataFormat	1 - 実線
DataMember	
DataSource	
DragIcon	(なし)
DragMode	0 - 手動

1 ここをクリック
2 ここをクリック
3 「1 ー実線」に設定

22 完成したMDIフォーム

解　説

1 　このプログラムの仕様について説明をしておきます。

まず、MDIフォームには、ツールバーコントロールとメニューコントロールを配置し、これらを使って表示するファイルを指定するようにします。ファイルの選択は、コモンダイアログコントロールを使用して行います。1つのコモンダイアログコントロールで、2種類のファイルを開く設定を行うのがポイントです。

最初のフォーム「Form1」には、リッチテキストボックスコントロールを配置し、文書ファイルを開くときは、このフォームのインスタンスを作成して表示するようにします。

「Form2」にはイメージコントロールを配置し、イメージファイルを開くときは、この「Form2」のインスタンスを作成して表示します。そのため、プロジェクトファイルには、2つのフォームを組み込んでいます。

表示する内容でフォームを使い分ける

2　プログラムのベースになるウィンドウは、MDIフォームにします。このMDIフォームには、ツールバーコントロールを配置します。

　また、メニューコントロールも組み込み、ファイルを開く操作はどちらからでもできるようにしています。このあたりは、通常のWindowsのアプリケーションを参考にしながら、自分で工夫してユーザーインターフェイスを作成してください。

［ウィンドウ］メニューは、複数の子フォームを表示したときの、ウィンドウの整列を行うためのメニューです。また、「ウィンドウリスト」のプロパティにチェックを付けているので、子フォームを作るたびに、子フォームの名前がリスト表示されます。

子フォームがタイル状に並んで、フォーム名がメニューにリスト表示される

3　ツールバーコントロールとイメージリストコントロールの操作方法は、「Lesson5 ツールバーコントロールの組み込み」で説明したとおりです。ツールバーのボタンの数が多い場合は、「キー」を設定して識別しやすくしたほうがいいでしょう。ここでは、3つのボタンしか使用していませんから、インデックスだけで管理しています。なお、ツールバーコントロールのボタンに、イメージと文字を組み込むと、どうしてもボタンのサイズが大きくなります。小さいボタンを使いたい場合は、イメージリストコントロールのイメージサイズを「16×16」にしたり、イメージ表示だけのボタンにすれば可能になります。

それぞれのボタンサイズの違い

まとめ

ユーザーインターフェイスを設計する場合は、一般的なアプリケーションの操作方法を踏襲することをお勧めします。あまり変わった操作方法を採用すると、初心者ユーザーなどは混乱してプログラムが操作できなくなってしまいますから、注意してください。

また、ツールバーコントロールのボタン表示は、ユーザーが見てわかるようなデザインを採用しましょう。「ツールヒント」プロパティを活用するのもひとつの手段です。

マルチドキュメントビューワ
ファイル　ウィンドウ
Exit　Text　Image
JPG、BMP、GIF形式のグラフィックファイルを開きます

ツールヒントでボタンの内容を説明する

練習問題

Q1　プログラム実行時に、ツールバーのボタンにマウスのポインタが重なると、ポインタが指の形に変わるように、プログラムを修正してください。

マルチドキュメントビューワ
ファイル　ウィンドウ
Exit　Text　Image
JPG、BMP、GIF形式のグラフィックファイルを開きます

ボタンの上でポインタが変わる

解答は巻末に→

Lesson 17

第５日
３時限目

マルチドキュメントビューワの作成

Lesson17.vbp
VB6CD2
Lesson17

コードの作成

Lesson16で作成したユーザーインターフェイスに、動作処理を行うコードを作成していきます。コードは、MDIフォームと標準フォーム「Form1」のそれぞれに作成します。MDI機能を持ったプログラムでは、フォームごとに動作を設定したい場合は、それぞれのフォームモジュールにコードを組み込む必要があります。

今回作成するプログラム

マルチドキュメントビューワ
ファイル　ウィンドウ
水平に並べて表示
縦に並べて表示
重ねて表示
1 C:¥WINDOWS¥デスクトップ¥VB6実践¥付録¥material¥window.jpg
✓2 C:¥Frunlog.txt
Exit

FirstRunScreens:Start
SpezialGeninstalls:Start
SpezialGeninstalls:Looking for
:C:¥WINDOWS¥OPTIONS¥PREDUP.TAG:
Intl:INTL:0:
GetUserInfo:INIT
SUMB:セットアップ:Windows をセットアップするには、ユーザー名を入力してください。:OK
VcpClose:About to close
VcpClose:About to End
VcpClose:About to Terminate
DoPreInstallWork:Auditmode :0:
PrepareRunonce:RUNONCE Section:RO_PREINSTALLAPP:
PrepareRunonce:Process :コントロール パネル: :1

個別のフォームに文書ファイルやイメージファイルなど、複数のファイルを表示できるプログラムを作成する

POINT

ツールバーコントロールや、メニューコントロールなど、MDIフォームに組み込んだコントロールの処理は、MDIフォームモジュールに作成します。また、Lesson15で説明したように、子ウィンドウの共通処理はインスタンスの元になるフォームのコードウィンドウに、処理を作成します。

今回は、コードの作成のみなので、コントロールとプロパティの設定はありません。

コードの記述

MDIForm1:Toolbar1-ButtonClick

Lesson17 - MDIForm1 (コード)

Toolbar1 ButtonClick

```
Private Sub Toolbar1_ButtonClick(ByVal Button As
MSComctlLib.Button)
    Select Case Button.Index
        Case 1
            Unload Me
            End
        Case 2
            Call OpenText
        Case 3
            Call OpenImage
    End Select
End Sub
```

MDIForm1:(General)-GetSource

Lesson17 - MDIForm1 (コード)

(General) GetSource

```
Public Function GetSource(ByVal SourceName As String) As String

    If SourceName = "TXT" Then
        With CommonDialog1
            .Filter = "文書ファイル(*.txt,*.rtf)| *.txt; *.rtf"
            .DialogTitle = "文書ファイルを開く"
            .filename = ""
             On Error GoTo FINISH
            .ShowOpen
            GetSource = .filename
        End With
    ElseIf SourceName = "IMG" Then
        With CommonDialog1
            .Filter = "Image file(*.bmp,*.gif,*.jpg)| " & _
                      "*.bmp;*.gif;*.jpg"
            .DialogTitle = "イメージファイルを開く"
            .filename = ""
            On Error GoTo FINISH
            .ShowOpen
            GetSource = .filename
        End With
```

```
    End If
    Exit Function

FINISH:
    GetSource = ""
End Function
```

MDIForm1:(General)-OpenText

Lesson17 - MDIForm1 (ｺｰﾄﾞ)

(General) | OpenText

```
Public Sub OpenText()
    Dim Fname As String

    Fname = GetSource("TXT")
    If Fname = "" Then
        Exit Sub
    End If
    Dim TextForm As New Form1
    With TextForm
        .Show
        .Caption = Fname
        .RichTextBox1.LoadFile Fname
        .RichTextBox1.Width = .Width - 700
        .RichTextBox1.Height = .Height - 700
    End With
End Sub
```

MDIForm1:(General)-OpenImage

Lesson17 - MDIForm1 (ｺｰﾄﾞ)

(General) | OpenImage

```
Public Sub OpenImage()
    Dim Fname As String

    Fname = GetSource("IMG")
    If Fname = "" Then
        Exit Sub
    End If
    Dim ImageForm As New Form2
    With ImageForm
        .Show
        .Caption = Fname
        .Image1.Picture = LoadPicture(Fname)
    End With
End Sub
```

MDIForm1:IDM_TEXT-Click

```
Lesson17 - MDIForm1 (コード)
IDM_TEXT    Click

Private Sub IDM_TEXT_Click()
    Call OpenText
End Sub
```

MDIForm1:IDM_IMAGE-Click

```
Lesson17 - MDIForm1 (コード)
IDM_IMAGE    Click

Private Sub IDM_IMAGE_Click()
    Call OpenImage
End Sub
```

MDIForm1:IDM_EXIT-Click

```
Lesson17 - MDIForm1 (コード)
IDM_EXIT    Click

Private Sub IDM_EXIT_Click()
    Unload Me
    End
End Sub
```

MDIForm1:IDM_CASCADE-Click

```
Lesson17 - MDIForm1 (コード)
IDM_CASCADE    Click

Private Sub IDM_CASCADE_Click()
    MDIForm1.Arrange vbCascade
End Sub
```

MDIForm1:IDM_HORI-Click

```
Lesson17 - MDIForm1 (コード)
IDM_HORI    Click

Private Sub IDM_HORI_Click()
    MDIForm1.Arrange vbTileHorizontal
End Sub
```

MDIForm1:IDM_VERT-Click

Lesson17 - MDIForm1 (コード)

IDM_VERT | Click

```
Private Sub IDM_VERT_Click()
    MDIForm1.Arrange vbTileVertical
End Sub
```

Form1:Form-Resize

Lesson17 - Form1 (コード)

Form | Resize

```
Private Sub Form_Resize()
    If Me.WindowState <> vbMinimized Then
        With RichTextBox1
            .Width = Me.Width - 700
            .Height = Me.Height - 700
        End With
    End If
End Sub
```

解 説

1 まず簡単にプログラムの動作を説明します。

①ユーザーが、ツールバーの［テキスト］ボタンまたは［ファイル］メニューの［開く］で［文書ファイル］をクリックすると、ファイル形式がテキストかRTFのファイルを開くためのダイアログボックスが表示されます。

文書ファイルを開く

ファイルの場所(I): (C:)

Acrobat3 / Aol30I / Kpcms / Mouse / My Documents / Program Files / Quickcmd / Realmode / temp / Windows / Yamaha / Frunlog.txt / Netlog.txt / Resetlog.txt / Setuplog.txt / test.rtf / 秀丸送金id.txt

ファイル名(N): 開く(O)

ファイルの種類(T): 文書ファイル(*.txt;*.rtf) キャンセル

読み取り専用ファイルとして開く(R)

②このコモンダイアログボックスは、ダイアログタイトルと「ファイルの種類」を設定して表示し、テキストかRTFファイル以外は選択できないようにしています。

③ファイルが選択されると、リッチテキストボックスコントロールを組み込んだ、標準フォーム「Form1」を元にインスタンスを作成し、子フォームとして表示します。

子フォームに表示されたファイル

④表示された子フォームは、そのファイル名をフルパスでフォームのキャプションに表示し、現在どのファイルを表示しているのかがわかるようにしています。

タイトルバーにファイルのパスが表示される

また、フォームのサイズとリッチテキストボックスコントロールのサイズを自動計算して調節しているため、フォームを拡大縮小しても、常に同じ比率でリッチテキストボックスコントロールも拡大縮小されて表示されます。

フォームのサイズが変わっても...

常に同じ比率でリッチテキストボックスコントロールも拡大縮小される

⑤MDI機能の特徴で、子フォームを全画面表示することもできます。

全画面表示

⑥ツールバーの［Image］ボタンまたは［ファイル］メニューの［開く］で［イメージファイル］をクリックすると、今度はイメージファイル用に設定されたコモンダイアログボックスが表示されます。やはり、ダイアログタイトルと「ファイルの種類」をコードで設定しており、ファイルの拡張子が「bmp」「gif」「jpg」のファイルだけが選択できるようにしています。

⑦ファイルが選択されると、イメージコントロールを組み込んだ標準フォーム「Form2」を元にインスタンスを作成して、イメージを表示します。

Form2を使ってイメージが表示される

⑧［ウィンドウ］メニューには、MDIフォーム内に表示した子フォームのウィンドウ状態を設定するサブメニューを組み込んであります。これらのメニューのClickイベントプロシージャには、複数のフォームのウィンドウ整列を変更するコードを組み込んであります。

上下に並べて表示

左右に並べて表示

重ねて表示

2 では、順番にコードを作成していきましょう。まずは、ツールバーコントロールのボタンがクリックされた場合の処理です。ツールバーコントロールのボタンがクリックされると、ButtonClickイベントプロシージャの引数「Button」には、クリックされたボタンオブジェクトが格納されます。この［Index］プロパティを把握することで、どのボタンがクリックされたのかがわかります。

これを、Select Caseステートメントで3つの処理に振り分けます。［Exit］ボタンは、そのままプログラムを終了させる処理をします。［Text］ボタンと［Image］ボタンは、それぞれのファイルを開いて表示するプロシージャ「OpenText」「OpenImage」を呼び出します。これらのプロシージャは、ユーザー定義のプロシージャとして作成し、ツールバーとメニューで1つの処理を呼び出せるようにしています。

```
Private Sub Toolbar1_ButtonClick(ByVal Button As _
                                    MSComctlLib.Button)
    Select Case Button.Index
        Case 1
            Unload Me
            End
        Case 2
            Call OpenText
        Case 3
            Call OpenImage
    End Select
End Sub
```

3 ここで、コモンダイアログコントロールの設定と、ファイル名を取得する処理を作成します。テキストファイルも、イメージファイルも、同じようにコモンダイアログボックスを表示して、ファイル名を取得するため、1つのプロシージャでテキストとイメージを使い分けられるようにすれば、処理が単純化できます。また、取得したファイル名を返してくるようにすれば、使いやすいプロシージャになります。

そこで、引数にテキストなのかイメージなのかを指定して実行すると、その種類にあったファイル名を返してくるような、ユーザー定義の「Function」プロシージャを作成することにします。「Function」とは、関数のように、引数と戻り値を持ったプロシージャを作成するステートメントです。「Function」のあとに、プロシージャと同様の規則で名前を付け、() に引数を設定します。そして、末尾に「As」を付けて戻り値のデータ型を指定します。

【書式】

```
[Public] Function name [(arglist)] [As type]
```

Public	Function プロシージャのスコープ（参照範囲）を指定します。Sub プロシージャと同じ規則です。「Public」はすべてのモジュールから参照可能で、「Private」は1つのモジュール内のすべてのプロシージャから参照可能になります。
name	Function プロシージャの名前です。
arglist	引数を記述します。引数付き Sub プロシージャと同様、値渡しの場合は「ByVal」を付け、参照渡しの場合は「ByRef」を付けて、引数名とデータ型を指定します。
As Type	Function が実行されて値を返してくるときの、戻り値のデータ型を指定します。

ここでは、「GetSource」という名前のFunctionプロシージャを作成し、引数に「SourceName」という文字列型の変数を値渡しで使用します。Functionプロシージャの実行結果は、ファイル名を文字列で返してくるようにするので、「As String」という戻り値のデータ型を指定します。

```
Public Function GetSource(ByVal SourceName As String) _
    As String
```

ワンポイントアドバイス

Functionプロシージャは、必ず「End Function」で終わらなければなりません。

4　では、実際にこの「GetSource」プロシージャを作成しましょう。最初に、引数「SourceName」の内容を、Ifステートメントで判断します。この引数は、テキストファイルを開くコモンダイアログボックスを表示する場合は、引数に「TXT」を設定するようにします。また、イメージファイル用のコモンダイアログボックスを表示する場合は、引数に「IMG」を指定します。

引数が「TXT」の場合は、コモンダイアログボックスに、ファイルの拡張子が「*.txt」「*.rtf」のファイルだけしか表示しないように、［Filter］プロパティを設定します。

```
If SourceName = "TXT" Then
    With CommonDialog1
        .Filter = "文書ファイル(*.txt,*.rtf)| *.txt; *.rtf"
```

次に、ダイアログボックスのタイトル文字列を設定し、「ファイル名」のところが空白になるようにします。

```
.DialogTitle = "文書ファイルを開く"
.filename = ""
```

これで、コモンダイアログボックスの各プロパティの設定は終了です。続けて「ShowOpen」メソッドで、［ファイルを開く］ダイアログボックスを表示します。

ユーザーがファイル名を選択すると、コモンダイアログボックスの［filename］プロパティに、フルパスでそのファイル名が格納されます。これを、このFunctionプロシージャとまったく同じ名前の変数に格納します。この変数が、Functionプロシージャが実行されたときの戻り値になります。

```
.ShowOpen
GetSource = .filename
```

すなわち、このプロシージャが実行され、ファイル名が選択されると、選択したファイル名をフルパスで、戻り値として返すようになります。

5 同じように、引数が「IMG」の場合のコモンダイアログボックスの設定を作成します。

```
ElseIf SourceName = "IMG" Then
    With CommonDialog1
        .Filter = "Image file(*.bmp,*.gif,*.jpg)| " & _
                  "*.bmp;*.gif;*.jpg"
        .DialogTitle = "イメージファイルを開く"
        .filename = ""
        .ShowOpen
        GetSource = .filename
    End With
End If
```

また、コモンダイアログボックスの［キャンセル］ボタンがクリックされた場合は、なんの処理もせずに終了します。この場合は、［filename］プロパティの値は、「""」（空白）のままです。

6 ファイル名を取得するFunctionプロシージャが作成できたら、実際にフォームのインスタンスを生成して、子フォームを表示する処理を作ります。

まず、テキストファイルを開く処理です。この処理は、ツールバーとメニューの両方で同じ処理が行えるように、ユーザー定義のSubプロシージャとして作成します。プロシージャ名は「OpenText」です。

```
Public Sub OpenText()
```

最初に、変数を1つ宣言します。先に作成したFunctionプロシージャ「GetSource」の戻り値を格納する変数「Fname」です。Functionプロシージャ「GetSource」の戻り値は文字列型なので、同じデータ型で宣言します。

```
Dim Fname As String
```

次に、Functionプロシージャ「GetSource」を、「TXT」という文字列を引数にして実行します。これで、テキストファイル用のコモンダイアログボックスを表示するコードが実行され、ユーザーが選択したファイル名を返してきます。戻り値は、変数「Fname」に格納します。

```
Fname = GetSource("TXT")
```

テキスト用のコモンダイアログボックス

7 コモンダイアログボックスの［キャンセル］ボタンがクリックされれば、戻り値は「""」が返されますから、これをIfステートメントで判断し、キャンセルであればここでプロシージャを終了します。

```
If Fname = "" Then
    Exit Sub
End If
```

ファイル名が返されていれば、リッチテキストボックスコントロールを組み込んだ標準フォーム「Form1」のインスタンスを、Dim...Newステートメントで作成します。作成されたオブジェクトは、「TextForm」という名前にします。

```
Dim TextForm As New Form1
```

そして、作成したインスタンスの子フォームを表示します。また、そのファイル名をフォームのキャプションに表示し、リッチテキストボックスコントロールの「LoadFile」メソッドに引き当てます。リッチテキストボックスコントロールは、ファイルの内容を読み込み表示します。

```
With TextForm
    .Show
    .Caption = Fname
    .RichTextBox1.LoadFile Fname
```

最後に、リッチテキストボックスコントロールの幅と高さを、子フォームの幅と高さの変更に連動させる処理を作成しておきます。理由は、「Form1」のコードで説明します。

```
.RichTextBox1.Width = .Width - 700
.RichTextBox1.Height = .Height - 700
```

これで、テキストファイルを表示する処理が完成です。

リッチテキストボックスは、フォームより1まわり小さく表示される

8

同様の方法で、イメージファイルを表示する処理を作成します。ユーザー定義のSubプロシージャ「OpenImage」を作成し、ファイル名を格納する変数「Fname」を宣言します。

```
Public Sub OpenImage()
    Dim Fname As String
```

次に、Functionプロシージャ「GetSource」を、引数に文字列「IMG」を設定して実行すれば、イメージファイル用のコモンダイアログボックスが表示されます。

```
Fname = GetSource("IMG")
```

イメージファイル用のコモンダイアログボックス

コモンダイアログボックスの［キャンセル］ボタンがクリックされれば、このプロシージャを終了します。

```
If Fname = "" Then
    Exit Sub
End If
```

ファイル名が返されれば、戻り値のファイル名を使用して、今度はイメージコントロールを組み込んでいるフォーム「Form2」を基にインスタンスを作成して、オブジェクト名を「ImageForm2」にします。そして、インスタンスの子フォームを表示し、フォームのキャプションにファイル名を設定します。また、イメージコントロールの［Picture］プロパティに、LoadPicture関数を使用してイメージファイルを設定し表示します。

```
Dim ImageForm As New Form2
With ImageForm
    .Show
    .Caption = Fname
    .Image1.Picture = LoadPicture(Fname)
End With
```

これで、イメージを表示するプロシージャは完成です。

9 すでに、ツールバーコントロールのイベントプロシージャでは、作成したテキストとイメージを表示するプロシージャを呼び出すコードを記述してありますから、あとはメニューコントロールのそれぞれのイベントプロシージャに設定して、メニューからも呼び出せるようにします。

```
Private Sub IDM_TEXT_Click()
    Call OpenText
End Sub

Private Sub IDM_IMAGE_Click()
    Call OpenImage
End Sub
```

また、［ファイル］メニューの［終了］のClickイベントプロシージャに、プログラムの終了処理を作成しておきます。

```
Private Sub IDM_EXIT_Click()
    Unload Me
    End
End Sub
```

10 次に、［ウィンドウ］メニューの、子フォームのウィンドウ表示を変更する処理を作成します。MDIフォームの［Arrange］プロパティを使うと、MDIフォーム内の子フォームのウィンドウ表示を変更できます。たとえば、

```
MDIForm1.Arrange vbCascade
```

と設定すると、MDIフォーム内のフォームはすべて重ねて表示されます。［Arrange］プロパティの値は、表示するウィンドウスタイルを定数で設定します。

定数	値	内容
vbCascade	0	重ねて表示します。
vbTileHorizontal	1	左右に並べて表示します。
vbTileVertical	2	上下に並べて表示します。
vbArrangeIcons	3	子フォームのアイコンを整列します。

この値を、それぞれのメニューのClickイベントプロシージャに設定すれば、メニューでウィンドウの表示を変えることができます。

```
Private Sub IDM_CASCADE_Click()
    MDIForm1.Arrange vbCascade
End Sub

Private Sub IDM_HORI_Click()
    MDIForm1.Arrange vbTileHorizontal
End Sub

Private Sub IDM_VERT_Click()
    MDIForm1.Arrange vbTileVertical
End Sub
```

11 MDIフォームの処理が完成したら、今度はテキストを表示するフォームに、リッチテキストボックスコントロールの表示サイズを自動調節する処理を組み込みます。インスタンスとして作成した子フォームは、ユーザーがウィンドウサイズを自由に変更できる「可変」フォームになっています。

また、子フォームは、全画面表示にできますから、フォームのサイズ変更にあわせてリッチテキストボックスコントロールのサイズも変更させないと、おかしな表示になってしまいます。そこで、リッチテキストボックスコントロールの［Width］と［Height］プロパティの値を、フォームの［Width］と［Height］プロパティの値を使って設定させれば、フォームのサイズが変わった時点で、自動的にリッチテキストボックスコントロールのサイズもそれに合わせて変更されるようになります。

サイズを調節しないと、おかしな表示になる

サイズを調節すると、きれいに表示される

テキスト用の子フォームは、フォーム「Form1」のイベントプロシージャに設定した処理がそのまま継承されます。また、フォームのサイズが変更されると、「Resize」というイベントが発生します。そこで、「Form1」の「Resize」イベントプロシージャに、リッチテキストボックスコントロールのサイズを自動調節するコードを作成すれば、インスタンスの子フォームにも適用されます。

まず、フォームの［WindowState］プロパティの値を調べ、子フォームが現在アイコン状態になっていないかどうかを調べます。アイコン状態でリッチテキストボックスコントロールの幅の調節を実行しようとするとエラーになってしまいますから、アイコン状態以外のときにこの処理を行うようにします。

あとは、フォームの［Width］と［Height］プロパティの値をリッチテキストボックスコントロールの［Width］と［Height］プロパティに設定します、このとき、それぞれ「-700」とマージンの分を差し引いて設定すれば、見た目がきれいなフォームになります。

```
Private Sub Form_Resize()
    If Me.WindowState <> vbMinimized Then
        With RichTextBox1
            .Width = Me.Width - 700
            .Height = Me.Height - 700
        End With
    End If
End Sub
```

このイベントプロシージャは、フォームのサイズが変更されれば常に呼び出されて実行されますから、フォームが最大化状態になっても、それに連動してリッチテキストボックスコントロールのサイズが拡大されるようになります。

まとめ

以上で、プログラムは完成です。コモンダイアログボックスを使って、テキストやイメージファイルをいくつでも開けるビューワが完成です。「ウィンドウ」メニューを使えば、選んだフォームをアクティブにして最前面に表示できます。

ここでは、リッチテキストボックスコントロールとイメージコントロールを使ったMDIフォームプログラムを作成しましたが、ほかにもフレキシブルグリッドコントロールを使うなど、コントロールとフォームの組み合わせでいろいろなプログラムが作成できます。

Functionプロシージャは、関数と同様の処理を自分で作成できます。引数を指定したり、実行結果を受け取ることができるので、汎用のプロシージャを作成する場合は、威力を発揮します。戻り値を返すには、プロシージャと同じ名前の変数を使う、というのがポイントです。

練習問題

Q1

テキスト用フォームとイメージ用フォームに、それぞれ別のアイコンを組み込んで表示するプログラムに修正してください。また、ステータスバーコントロールを組み込み、時刻表示をするようにしてください。

なお、アイコンは、すでにMDIフォームに配置しているイメージリストコントロールから使用してください。

解答は巻末に→

V i s u a l B a s i c 6

Let's VBSchool!!

第6日 (Lesson18～Lesson21)
インターネットプログラムの作成

L e a r n i n g S c h o o l

Visual Basic 6.0に付属のActiveXコントロールに、「Microsoft Internet Transfer Control 6.0」（インターネットトランスファーコントロール）があります。これは、インターネットのプロトコルである「HTTP」と「FTP」プロトコルを使って、サーバ・クライアント間のデータ転送を行います。今日のレッスンでは、このコントロールを使って、WebにあるHTMLを自動ダウンロードするプログラムとFTPプロトコルでサーバのディレクトリを取得するプログラムを作成します。

今日のレッスンで解説する項目

- ❏ インターネットトランスファーコントロールの機能
- ❏ HTTPプロトコルでの転送方法
- ❏「Execute」メソッドを使ったHTMLファイルのダウンロード
- ❏「StateChanged」イベントの役割と使い方
- ❏「GetChunk」メソッドでの転送データの取り出し
- ❏ Shell関数で他のプログラムを起動する
- ❏ FTPプロトコルのコマンドの使用方法
- ❏ リストビューコントロールの機能
- ❏ ビューの切り替え方法
- ❏ アイコンの組み込みと項目名の設定
- ❏ Reportビューでのコラムヘッダーの設定
- ❏ サブアイテムの追加
- ❏ クリックされた項目の取得方法
- ❏ リストビューコントロールを使ったディレクトリ情報の表示
- ❏ プログレスバーを使ったダウンロード状況の表示
- ❏「GET」コマンドによるファイルのダウンロード
- ❏ コモンダイアログボックスを使ったファイル転送方法

Lesson
18

第６日
１時限目

インターネットプログラムの作成

Lesson18.vbp
VB6CD2
Lesson18

ホームページダウンロードプログラムの作成

最初は、インターネットに接続した状態で、指定したURLのHTMLをダウンロードし、HTMLテキストをリッチテキストボックスコントロールに表示するプログラムを作成します。ダウンロードしたHTMLは、テンポラリファイルとして保存され、インターネットエクスプローラを使って表示させます（ただし、テキストのみでグラフィックは保存しません）。

今回作成するプログラム

Form1

Get!　Save　View　Exit

```
<html>

<head>
<meta http-equiv="Content-Type"
content="text/html; charset=x-sjis">
<meta name="GENERATOR" content="Microsoft FrontPage 4.0">
<title>Information</title>
</head>

<body bgcolor="#DABADA">
<div align="center"><center>

<table border="2">
    <tr>
        <td><font color="#800080" size="6">Information</font></td>
    </tr>
</table>
</center></div>

<p align="center"> </p>
<div align="center">
  <center>

<table border="2" cellpadding="8" width="85%">
```

ダウンロードしたHTMLテキストをここに表示する

POINT

インターネットトランスファーコントロールを使って、HTTPプロトコルで指定したURLのWebサーバにアクセスし、HTMLをダウンロードします。ダウンロードしたHTMLドキュメントは、フォームに配置したリッチテキストボックスコントロールで表示します。また、Shell関数を使用してインターネットエクスプローラを呼び出し、HTMLを表示します。

コントロールとプロパティの設定

※新しいプロジェクトファイル（標準EXE）を用意します。

1 ツールパレットに次のコンポーネントを追加する

- Microsoft Internet Transfer Control 6.0
- Microsoft Rich Textbox Control 6.0
- Microsoft Windows Common Controls 6.0

ツールパレットにコモンコントロールが追加される

2 フォームを広げ、ツールパレットの［StatusBar］を使って、ステータスバーを配置する

Form1

1 斜めにドラッグして配置

コンポーネントを追加する

コンポーネントを追加する方法については、「Lesson5 ツールバーコントロールの組み込み」を参照してください。

[プロパティページ] ダイアログ

[プロパティページ] ダイアログボックスを表示するには、コントロールを選択し、プロパティウィンドウの [(プロパティページ)] をクリックし、右欄の [...] ボタンをクリックします。

コマンドボタンを配置する

コマンドボタンを配置するには、ツールパレットの [CommandButton] を使います。

3 [プロパティページ] の [パネル] タブで、[サイズの自動設定] を「1 – sbrSpring」に設定する

4 フォームにコマンドボタンコントロールを4つ配置し、各ボタンのプロパティを次のように設定する

コントロール	オブジェクト名	Caption
Command1	GetBtn	Get!
Command2	SaveBtn	Save
Command3	ViewBtn	View
Command4	ExitBtn	Exit

5 ツールパレットの [Inet] を使って、インターネットトランスファーコントロールを配置する

6 ツールパレットの［RichTextBox］を使って、リッチテキストボックスコントロールを配置する

1 斜めにドラッグして配置

コントロールの重なり

このとき、インターネットトランスファーコントロールが隠れていますが、このコントロールはプログラム実行時にはフォーム上に表示されませんので、重なっていてもだいじょうぶです。

7 ［ScrollBars］プロパティを「3 – rtfBoth」に設定する

1 ここをクリック

2 ここをクリック

3 ここをクリック

8 ［Text］プロパティを「ここにHTMLの内容が表示されます。」に変更する

1 ここに入力

9 完成したユーザーインターフェイス

Form1

Get! Save View Exit

ここにHTMLの内容が表示されます。

コードの記述

Form1:GetBtn-Click

Lesson18 - Form1 (ｺｰﾄﾞ)

GetBtn / Click

```
Private Sub GetBtn_Click()
    Dim GetUrl As String

    StatusBar1.Panels(1).Text = "Run"
    GetUrl = "http://www.big.or.jp/~seto/info.htm"
    Inet1.Execute GetUrl, "GET"
End Sub
```

Form1:Inet1-StateChanged

Lesson18 - Form1 (ｺｰﾄﾞ)

Inet1 / StateChanged

```
Private Sub Inet1_StateChanged(ByVal State As Integer)
    Dim GetData As Variant
    If State = icResponseCompleted Then
        MsgBox "Completed"
        GetData = Inet1.GetChunk(5000, icString)
        RichTextBox1.Text = GetData
        StatusBar1.Panels(1).Text = ""
    End If
End Sub
```

Form1:Savebtn-Click

Lesson18 - Form1 (コード)

Savebtn | Click

```
Private Sub Savebtn_Click()
    Call SaveHtml
End Sub
```

Form1:ViewBtn-Click

Lesson18 - Form1 (コード)

ViewBtn | Click

```
Private Sub ViewBtn_Click()
    Call SaveHtml
    Shell "C:¥Program Files¥Internet Explorer¥IEXPLORE.EXE" & _
        " C:¥temp.htm", vbNormalFocus
End Sub
```

Form1:(General)-SaveHtml

Lesson18 - Form1 (コード)

(General) | SaveHtml

```
Private Sub SaveHtml()
    Open "c:temp.htm" For Output As #1
    Print #1, RichTextBox1.Text
    Close #1
End Sub
```

Form1:ExitBtn-Click

Lesson18 - Form1 (コード)

ExitBtn | Click

```
Private Sub ExitBtn_Click()
    Unload Me
    End
End Sub
```

Form1:Form-Resize

Lesson18 - Form1 (コード)

Form | Resize

```
Private Sub Form_Resize()
    If WindowState <> vbMinimized Then
        With RichTextBox1
            .Width = Form1.Width - 500
            .Height = Form1.Height - 1500
        End With
    End If
End Sub
```

解説

1

インターネットトランスファーコントロールは、インターネットの2つのプロトコルである「HTTP」と「FTP」プロトコルを操作するコントロールです。「HTTP」プロトコルは、ホームページで有名なWWW（World Wide Web）サーバにアクセスし、指定したHTMLファイルの読み込みなどを行うことができます。

このプログラムでは、インターネットトランスファーコントロールで「HTTP」プロトコルを操作し、特定のURLのHTMLファイルをダウンロードします。ダウンロードしたHTMLファイルは、いったんバッファに格納されますので、それを読み出してリッチテキストボックスコントロールに表示します。

では、簡単にプログラムの動作を説明します。

①あらかじめ、コンピュータをインターネットに接続しておきます。

ワンポイントアドバイス

イントラネットの場合は、ファイアウォールの設定によっては、HTMLドキュメントのダウンロードができない場合があります。その際は、ネットワークのシステム管理者にプロキシサーバーのアドレスなどを問い合わせてください。

②プログラムは、URLを筆者のホームページに指定しています。

```
http://www.big.or.jp/~seto/info.htm
```

プログラムを起動し［Get］ボタンをクリックすると、指定したHTMLファイルをダウンロードします。

ここをクリックすると、HTML ファイルがダウンロードされる

③転送が終了すると、メッセージボックスが表示されます。

④メッセージボックスの［OK］ボタンをクリックすると、リッチテキストボックスコントロールに、ダウンロードしたHTMLの内容が表示されます。

ダウンロードした HTML の内容

⑤［Save］ボタンをクリックすると、ダウンロードしたHTMLの内容を、「c:¥temp.htm」という名前でテンポラリファイルを作成し保存します。

⑥［View］ボタンをクリックすると、インターネットエクスプローラを起動し、このテンポラリファイルをブラウザに表示します。

2

では、コードを作成していきましょう。最初に、［Get］ボタンの処理です。
変数「GetUrl」を宣言します。この変数には、これからアクセスするホームページのURLを格納します。次に、ステータスバーコントロールに「Run」という文字を表示し、これからダウンロードを開始することを表示します。

```
Private Sub GetBtn_Click()
    Dim GetUrl As String

    StatusBar1.Panels(1).Text = "Run"
```

また、変数「GetUrl」に、URLを格納します。

```
GetUrl = "http://www.big.or.jp/~seto/info.htm"
```

3

HTTPプロトコルを実行するには、インターネットトランスファーコントロールの「Execute」メソッドを使用します。このメソッドは、引数にHTTPプロトコルのコマンドを指定すると、そのコマンドを実行してくれます。Executeメソッドがサポートしている HTTPプロトコルのコマンドは、次ページのとおりです。
ここでは、指定したURLのHTMLファイルをダウンロードしますので、「GET」コマンドを指定します。

```
Inet1.Execute GetUrl, "GET"
```

【Executeメソッドの書式】

```
object.Execute [url[, operation[, data[, requestHeaders]]]]
```

object	対象となるオブジェクト（インターネットトランスファーコントロールなど）を指定します。
url	コントロールの接続先となるURLを指定します。省略可能。URLを指定しなかった場合は、URLプロパティのURLが使用されます。
operation	実行するコマンドを指定します。省略可能。サポートされているHTTPプロトコルのコマンドについては、下の表を参照してください。
data	省略可能。操作に使用するデータを文字列で指定します。
requestHeaders	省略可能。リモートサーバーから送信される追加ヘッダーを文字列で指定します。

＜HTTPプロトコルのコマンド＞

操作	内容
GET	指定したURL、またはURLプロパティに設定されているURLからデータを取得します。
HEAD	要求ヘッダーを送信します。
POST	データをサーバーへポストします。データは、引数dataで指定します。この操作ではGETと同じことができますが、この場合は、引数dataに追加命令を指定します。
PUT	Put操作。置き換えるページの名前は、引数dataで指定します。

これで、著者のWWWサーバから「info.htm」というファイルをダウンロードしてきます。なお、このExecuteメソッドは、非同期でサーバとの接続を行うため、転送が行われている間も、他の処理を実行できます。

4 「GET」コマンドを開始すると、インターネットトランスファーコントロールには「StateChanged」というイベントが発生します。このイベントプロシージャは、引数「State」を持ち、現在コントロールがどういう状態になっているのかを定数で格納しています。つまり、このプロシージャを利用することで通信状況を正しく把握することができます。

HTMLファイルのダウンロード中、あるいはダウンロード完了時に、プログラムになにか処理をさせたい場合は、このイベントプロシージャを活用します。ここでは、ダウンロードが完了したら、メッセージボックスでユーザーに知らせ、HTMLファイルをリッチテキストボックスコントロールに表示する処理を行うようにさせてみましょう。

まず、ダウンロードした内容を格納する変数「GetData」をバリアント型で宣言します。

```
Private Sub Inet1_StateChanged(ByVal State As Integer)
     Dim GetData As Variant
```

次に、Ifステートメントで、イベントプロシージャの引数「State」の内容を判断します。

ここでは、ダウンロードが完了した時点で、HTMLの内容をリッチテキストボックスコントロールに表示するようにしますから、すべてのデータが受信完了したかどうか（すなわち「State」が「icResponseCompleted」のとき）を調べます。

```
If State = icResponseCompleted Then
```

引数「State」に格納される定数については、以下の「Column」を参照してください。

COLUMN ・・・・・

StateChangedイベントプロシージャの引数Stateに格納される定数

定数	値	内容
icNone	0	状態についての報告はありません。
icResolvingHost	1	コントロールは、指定されたホストコンピュータのIPアドレスを調べています。
icHostResolved	2	コントロールは、指定されたホストコンピュータのIPアドレスを見つけました。
icConnecting	3	コントロールは、ホストコンピュータとの接続処理を行っています。
icConnected	4	コントロールは、ホストコンピュータと正常に接続しました。
icRequesting	5	コントロールは、ホストコンピュータに要求を送信しています。
icRequestSent	6	コントロールは、要求を正常に送信しました。
icReceivingResponse	7	コントロールは、ホストコンピュータからの応答を受信しています。
icResponseReceived	8	コントロールは、ホストコンピュータからの応答を正常に受信しました。
icDisconnecting	9	コントロールは、ホストコンピュータとの接続を解除しています。
icDisconnected	10	コントロールは、ホストコンピュータとの接続を正常に解除しました。
icError	11	ホストコンピュータとの通信中にエラーが発生しました。
icResponseCompleted	12	要求処理が完了し、すべてのデータが受信されました。

5 受信が完了したら、メッセージボックスで完了したことを示します。

```
MsgBox "Completed"
```

Lesson18

Completed

OK

メッセージの表示

そして、受信バッファにあるHTMLの内容を変数に格納します。HTTPプロトコルのGETコマンドで取得してきたHTMLファイルの内容は、いったんバッファに格納されます。そこで、「GetChunk」メソッドを使用して、バッファから内容を取り出します。「GetChunk」メソッドは、引数にバッファから取り出すデータのサイズと、文字列で取得するのか、バイト配列で取り出すのかを指定します。

Executeメソッドでダウンロードを指定したHTMLファイルは、約5KBのサイズなので、ここではデータサイズを「5000」にし、文字列で取り出しています。

```
GetData = Inet1.GetChunk(5000, icString)
```

【GetChunk メソッドの書式】

```
object.GetChunk(size [,datatype])
```

object	オブジェクトを指定します。
size	取得するチャンク（ひとまとまりのデータ）のサイズを長整数で指定します。
datatype	取得するチャンクのデータ型を整数で指定します。

＜datatypeの定数＞

定数	値	内容
icString	0	データを文字列として取得します。既定値です。
icByteArray	1	データをバイト配列として取得します。

受信バッファの内容を変数に格納したら、これをリッチテキストボックスコントロールの［Text］プロパティに代入します。また、「Run」と表示させていたステータスバーコントロールの文字列を削除します。

```
RichTextBox1.Text = GetData
StatusBar1.Panels(1).Text = ""
```

これで、HTMLファイルのダウンロード処理が完了しました。

Form1

Get!　Save　View　Exit

```
<html>

<head>
<meta http-equiv="Content-Type"
content="text/html; charset=x-sjis">
<meta name="GENERATOR" content="Microsoft FrontPage 4.0">
<title>Information</title>
</head>

<body bgcolor="#DABADA">
<div align="center"><center>

<table border="2">
    <tr>
        <td><font color="#800080" size="6">Information</font></td>
    </tr>
</table>
</center></div>

<p align="center"> </p>
<div align="center">
  <center>

<table border="2" cellpadding="8" width="85%">
```

HTMLの内容が表示される

6 ［Save］ボタンがクリックされたときは、リッチテキストボックスコントロールの内容をファイルに書き出します。この処理は、独自のユーザー定義プロシージャ「SaveHtml」を作成し、これを呼び出します。

```
Private Sub Savebtn_Click()
    Call SaveHtml
End Sub
```

ユーザー定義プロシージャ「SaveHtml」は、最初に「Open」メソッドで「c:¥temp.htm」というテンポラリファイルを、シーケンシャルのテキストモードで作成します。そして、ここに「Print #」ステートメントで書き込みます。これで、原文どおりにHTMLファイルを再現して保存できます。書き込んだら、Closeメソッドでファイルを閉じます。

```
Private Sub SaveHtml()
    Open "c:temp.htm" For Output As #1
    Print #1, RichTextBox1.Text
    Close #1
End Sub
```

7 ［View］ボタンは、Shell関数を使用して、インターネットエクスプローラを起動します。このとき、最新のHTMLを表示するために、一度リッチテキストボックスコントロールの内容をテンポラリファイルに保存するために、ユーザー定義プロシージャ「SaveHtml」を呼び出し、実行します。

Shell関数は、引数に起動するプログラム名をフルパスで指定し、起動時のファイルがあれば2番目の引数に指定します。また、プログラム起動時のウィンドウ表示状態を3番目に定数で指定します。ここでは、インターネットエクスプローラを起動プログラムに指定し、保存したテンポラリファイルを起動時のファイル指定にして、ウィンドウは標準の表示状態でShell関数を実行しています。

```
Private Sub ViewBtn_Click()
    Call SaveHtml
    Shell "C:¥Program Files¥Internet Explorer¥IEXPLORE.EXE" & _
        " C:¥temp.htm", vbNormalFocus
End Sub
```

【Shell関数の書式】

```
Shell(pathname[,windowstyle])
```

pathname　実行するプログラム名をバリアント型の値で指定します。

windowstyle　実行するプログラムのウィンドウの形式を定数で指定します。省略すると、プログラムはフォーカスを持った状態で最小化表示されます。

＜windowstyleの定数＞

定数	値	内容
vbHide	0	非表示になるウィンドウ（フォーカス有り）
vbNormalFocus	1	通常のウィンドウ（フォーカス有り）
vbMinimizedFocus	2	最小化表示されるウィンドウ（フォーカス有り）
vbMaximizedFocus	3	最大化表示されるウィンドウ（フォーカス有り）
vbNormalNoFocus	4	フォーカスを持たないウィンドウ。現在アクティブなウィンドウはそのままです。
vbMinimizedNoFocus	6	最小化表示でフォーカスを持たないウィンドウ。現在アクティブなウィンドウはそのままです。

！ ワンポイントアドバイス

「フォーカス」というのは、ユーザーがマウスやキーボードを使って、イベントが起こったときに、オブジェクトがそのイベントを受け取れる状態にあることを指します。つまり、テキストを入力したりするときにはそのコントロールがユーザーにとって入力可能状態であることを、「フォーカスを持っている」といいます。ウィンドウがフォーカスを持った場合、そのウィンドウは再前面にあるということになります。

インターネットエクスプローラが起動し「temp.htm」が表示される。

8 最後に［Exit］ボタンで、プログラムの終了処理を行います。

```
Private Sub ExitBtn_Click()
    Unload Me
    End
End Sub
```

また、フォームのサイズがユーザーによって変更された時は、自動的にリッチテキストボックスコントロールのサイズも変更するように、サイズ調整の処理をフォームの「Resize」イベントプロシージャに作成して完了です。

```
Private Sub Form_Resize()
    If WindowState <> vbMinimized Then
        With RichTextBox1
            .Width = Form1.Width - 500
            .Height = Form1.Height - 1500
        End With
    End If
End Sub
```

まとめ

インターネットトランスファーコントロールでHTTPプロトコルによる非同期受信を行った場合は、「StateChanged」イベント内で「GetChunk」メソッドを実行して、受信バッファからダウンロードした内容を取得してください。ただし、このレッスンのプログラムではテキスト（HTMLファイル）だけをダウンロードしていますから、ホームページにリンクされている画像データ等はダウンロードされていません。

インターネットトランスファーコントロールの機能を使ったこのプログラムを発展させて、例えばWebの自動巡回プログラムなどを作成してみるのもいいでしょう。

このプログラムでは、HTMLファイルのダウンロードを行いましたが、イメージファイルなどは次のレッスンで紹介するFTPプロトコルによる転送を使用してください。

また、このプログラムを実行すると、作業用の一時ファイルとしてCドライブのルートディレクトリに「temp.htm」を作成します。必要なければプログラム終了後削除しておきましょう。

練習問題

Q1　フォームにテキストボックスコントロールを組み込み、ダウンロードするHTMLファイルを自由に指定できるようにしてください。

解答は巻末に→

ワンポイントアドバイス

インターネットトランスファーコントロールを使って、HTTPプロトコルで通信する場合は、Executeメソッド以外に「OpenURL」メソッドを使用できます。ただし、このメソッドは、同期転送を行うため、サーバへの接続を開始すると、他の操作ができません。また、トラフィックの混雑などで転送に時間がかかる場合は、すべてのドキュメントを受信しないままリッチテキストボックスコントロールへの表示が行われてしまう場合があります。できれば、非同期で動作するExecuteメソッドを使うことをお勧めします。

その他、イントラネットでプロキシサーバー経由でインターネットへ常時接続している場合、プログラムの実行は問題ありませんが、プログラムの終了に1分ぐらい待たされることがあるのでご注意ください。モデムで、インターネットプロバイダに直接接続している場合は、問題なくすぐに終了します。

Lesson 19

第6日
2時限目

インターネットプログラムの作成

Lesson19.vbp
VB6CD2
Lesson19

FTPプロトコルを使う

今度は、インターネットトランスファコントロールを使って、FTPプロトコルを操作します。FTPプロトコルは、インターネットのファイルを転送するプロトコルで、サーバのディレクトリ操作やファイルの操作を行うコマンドを持っています。このレッスンでは、FTPプロトコルを使って、Webサーバのディレクトリ情報を取得するプログラムを作成します。

今回作成するプログラム

POINT

インターネットトランスファコントロールの、FTPプロトコル転送機能を使って、指定したアノニマスFTPサーバに匿名でログインしてディレクトリ情報を取得します。取得したディレクトリは、リッチテキストボックスコントロールに表示します。

また、FTPサイトへのアクセスは、コンボボックスコントロールを使用して、リスト選択と入力の両方ができるようにします。

コントロールとプロパティの設定

※新しいプロジェクトファイル（標準 EXE）を用意します。

1 ツールパレットに次のコントロールを組み込む

- Microsoft Internet Transfer Control 6.0
- Microsoft Rich Textbox Control 6.0

ツールパレットにコンポーネントが追加される

2 フォームを広げ、フォームの［Caption］プロパティを「Mini FTP Cliant」に変更する

プロパティ - Form1	
Form1 Form	
全体	項目別
(ｵﾌﾞｼﾞｪｸﾄ名)	Form1
Appearance	1 - 3D
AutoRedraw	False
BackColor	&H8000000F&
BorderStyle	2 - 可変
Caption	Mini FTP Cliant
ClipControls	True
ControlBox	True

1 ここを変更

3 フォームにラベルコントロールを配置し、［Caption］プロパティを変更する

Mini FTP Cliant

URLの入力: 例　ftp.microsoft.com

［Caption］プロパティを「URLの入力: 例 ftp.microsoft.com」に変更したラベルコントロール

コンポーネントを追加する

コンポーネントを追加する方法については、「Lesson5 ツールバーコントロールの組み込み」を参照してください。

ラベルコントロールを配置する

ラベルコントロールを配置するには、ツールパレットの［Label］ A を使います。

4 ツールパレットの［ComboBox］を使って、コンボボックスコントロールを配置する

Mini FTP Cliant
URLの入力：例 ftp.microsoft.com
Combo1

1 斜めにドラッグして配置

5 ［Text］プロパティを「ftp.microsoft.com」に変更する

ﾌﾟﾛﾊﾟﾃｨ - Combo1
Combo1 ComboBox
全体 | 項目別

OLEDropMode	0 - なし
RightToLeft	False
Sorted	False
Style	0 - ﾄﾞﾛｯﾌﾟﾀﾞｳﾝ ｺﾝﾎﾞ
TabIndex	4
TabStop	True
Tag	
Text	ftp.microsoft.com
ToolTipText	
Top	540

1 ここを変更

6 ［List］プロパティのリストに項目を設定する（例：ftp.microsoft.com、ftp.borland.com）

ﾌﾟﾛﾊﾟﾃｨ - Combo1
Combo1 ComboBox
全体 | 項目別

Index	
IntegralHeight	True
ItemData	(ﾘｽﾄ)
Left	150
List	(ﾘｽﾄ)
Locked	ftp.microsoft.com
MouseIcon	ftp.borland.com
MousePointer	
OLEDragMode	
OLEDropMode	

List

1 ここをクリック
2 ここをクリック
3 ここに項目を設定

［List］プロパティの項目

［List］プロパティに設定した項目は、コンボボックスのドロップダウンリストとして一覧表示されます。

7 ツールパレットの［RichTextBox］を使って、リッチテキストボックスコントロールを配置する

Mini FTP Cliant

URLの入力：例 ftp.microsoft.com

ftp.microsoft.com

RichText

1 斜めにドラッグして配置

8 リッチテキストボックスの［ScrollBars］プロパティを「3 – rtfBoth」に設定する

プロパティ – RichTextBox1

RichTextBox1 RichTextBox

全体 | 項目別

OLEDropMode	2 – rtfOLEDropAuto
RightMargin	0
ScrollBars	3 – rtfBoth
TabIndex	0 – rtfNone
TabStop	1 – rtfHorizontal
Tag	2 – rtfVertical
	3 – rtfBoth
Text	RichText
ToolTipText	
Top	1065
Visible	True

1 ここをクリック

2 ここをクリック

3 ここをクリック

9 ツールパレットの［Inet］を使って、インターネットトランスファコントロールを配置する

Mini FTP Cliant

URLの入力：例 ftp.microsoft.com

ftp.microsoft.com

RichText

1 斜めにドラッグして配置

コマンドボタンを配置する

コマンドボタンを配置するには、ツールパレットの［CommandButton］を使います。

10 フォームにコマンドボタンを2つ配置し、各ボタンのプロパティを次のように設定する

コントロール名	オブジェクト名	Caption
Command1	GetBtn	Get Dir List
Command2	ExitBtn	Exit

Mini FTP Cliant
URLの入力：例 ftp.microsoft.com
ftp.microsoft.com
Get Dir List
Exit
RichText

設定したコマンドボタンコントロール

コードの記述

Form1:GetBtn-Click

Lesson19 - Form1 (ｺｰﾄﾞ)
GetBtn　Click

```
Private Sub GetBtn_Click()
    Dim GetUrl As String

    GetUrl = Combo1.Text
    GetUrl = "ftp://" & GetUrl
    Inet1.Execute GetUrl, "DIR"
End Sub
```

Form1:Inet1-StateChanged

Lesson19 - Form1 (ｺｰﾄﾞ)

Inet1 / StateChanged

```
Private Sub Inet1_StateChanged(ByVal State As Integer)
    Dim GetData As Variant
    If State = icResponseCompleted Then
        GetData = Inet1.GetChunk(2048, icString)
        RichTextBox1.Text = GetData
        MsgBox "ダウンロード完了"
    ElseIf State = icError Then
        MsgBox "ダウンロードできません"
    End If
End Sub
```

Form1:ExitBtn-Click

Lesson19 - Form1 (ｺｰﾄﾞ)

ExitBtn / Click

```
Private Sub ExitBtn_Click()
    On Error Resume Next
    Inet1.Execute , "CLOSE"
    Unload Me
    End
End Sub
```

ワンポイントアドバイス

プロジェクトを起動してそのまま終了しようとすると、下図のようなエラーがでます。これはリモートコンピュータへの接続が確立されていないにも関わらず、「CLOSE」コマンドを実行したためです。このエラーを防ぐために、「ExitBtn_Click」プロシージャの１行目に「On Error Resume Next」ステートメントを組み込んでおきます。

解説

1 このプログラムは、前のレッスンとほぼ同様の方法で、FTPプロトコルを使って、指定したFTPサーバのディレクトリ情報を取得するものです。プログラムの動作そのものはいたって簡単で、FTPプロトコルの「DIR」コマンドでサーバのディレクトリ情報を取得し、その内容をリッチテキストボックスコントロールに表示します。

①コンピュータをインターネットにモデムで直接接続し、プログラムを起動します。

②コンボボックスコントロールのリストからURLを選ぶか、直接入力して［Get Dir List］ボタンをクリックします。サーバに接続して、FTPプロトコルの「DIR」コマンドを実行します。

③FTPプロトコルのコマンドが実行されて、ダウンロードが完了すると、メッセージボックスが表示され、リッチテキストボックスコントロールにディレクトリの一覧が表示されます。

ワンポイントアドバイス

インターネットトランスファコントロールでは、プロキシサーバーを経由してインターネットに接続している場合、プロキシが「CERN プロキシサーバー」だと、Executeメソッドでダイレクトに FTP 接続することはできません（前レッスンのようにHTTPプロトコルは可能ですが)。「OpenURL」メソッドを使えますが、インターネットでこのコントロールを使った同期接続には安定性に問題があるので、基本的にはプロキシ経由での FTP のダイレクト接続はできないと認識しておいた方がよいでしょう。

2 では、コードを作成します。Executeメソッドの実行は、［Get Dir List］ボタンのClickイベントプロシージャに作成します。

まず、URLを格納する変数「GetUrl」を、文字列型で宣言します。URLは、コンボボックスコントロールの［Text］プロパティから取得します。

```
Private Sub GetBtn_Click()
    Dim GetUrl As String
    GetUrl = Combo1.Text
```

また、FTPプロトコルの場合は、URLの先頭にプロトコルの記号「ftp://」が付きますから、この文字列と結合してExecuteメソッドを実行します。サーバのディレクトリを取得する、FTPプロトコルのコマンドは、「DIR」になりますので、この文字をExecuteメソッドの引数に指定します。

```
GetUrl = "ftp://" & GetUrl
Inet1.Execute GetUrl, "DIR"
```

3 インターネットトランスファコントロールとサーバ間の転送が始まると、「StateChanged」イベントが発生します。転送が無事完了すれば、このイベントプロシージャの引数「State」に、受信完了の定数「icResponseCompleted」が格納されますから、バッファの内容を「GetChunk」メソッドで取り出し、リッチテキストボックスコントロールに転送します。そして、転送完了のメッセージボックスを表示します。この処理は、前のレッスンで行った方法と同じです。

```
Private Sub Inet1_StateChanged(ByVal State As Integer)
    Dim GetData As Variant

    If State = icResponseCompleted Then
        GetData = Inet1.GetChunk(2048, icString)
        RichTextBox1.Text = GetData
        MsgBox "ダウンロード完了"
```

転送に失敗すれば、「State」には「icError」が格納されますから、受信失敗のメッセージを表示して、このプロシージャを終了します。

```
ElseIf State = icError Then
    MsgBox "ダウンロードできません"
End If
```

4 最後に、［終了］ボタンの処理を作成します。FTPの場合は、Executeメソッドで「CLOSE」コマンドを実行し、FTPサーバへの接続を切っておきます。あとは、通常のプログラム終了処理を記述します。

```
Private Sub ExitBtn_Click()
    On Error Resume Next
    Inet1.Execute , "CLOSE"
    Unload Me
    End
End Su
```

Executeメソッドで URLを切り替えた場合は、自動的に先に接続されていたFTPサーバを切断してから再接続されるので、「CLOSE」コマンドは必要ありません。

COLUMN ……

Executeメソッドを FTP プロトコルで使う場合に使用できるコマンド

操作	内容
CD	file1 ディレクトリを変更します。file1 に指定されたディレクトリに移動します。
CDUP	親ディレクトリに移動します。"CD.." を実行した場合と同じ結果となります。
CLOSE	現在の FTP 接続を閉じます。
DELETE file1	file1 に指定されたファイルを削除します。
DIR file1	file1 に指定されたディレクトリ内を調べます（Directory)。ワイルドカードも指定できますが、構文を解釈するのはリモートホストです。file1 を指定しないと、現在のディレクトリ内のすべてのファイルとディレクトリの一覧が返されます。この一覧データを取得するには、GetChunk メソッドを使用します。
GET file1 file2	file1 に指定されたリモートファイルを取得し、file2 に指定された名前のローカルファイルを作成します。
LS file1	file1 に指定されたディレクトリ内を調べます（List)。ワイルドカードも指定できますが、構文を解釈するのはリモートホストです。ファイルの一覧データを取得するには、GetChunk メソッドを使用します。
MKDIR file1	file1 に指定されたディレクトリを作成します（Make Directory)。この操作が成功するかどうかは、ユーザーが持っているリモートホスト上でのユーザー特権に依存します。
PUT file1 file2	file1 に指定されたローカルファイルをリモートホストにコピーして、file2 に指定された名前のリモートファイルを作成します。
PWD	現在のディレクトリの名前を返します（Print Working Directory)。返されたデータを取得するには、GetChunk メソッドを使用します。
QUIT	現在のユーザーでの処理を終了します。
RECV file1 file2	file1 に指定されたリモートファイルを取得し、file2 に指定された名前のローカルファイルを作成します。GET と同じです。
RENAME file1 file2	file1 に指定されたリモートファイルの名前を file2 で指定された新しい名前に変更します。この操作が成功するかどうかは、ユーザーが持っているリモートホスト上でのユーザー特権に依存します。
RMDIR file1	file1 に指定されたリモートディレクトリを削除します（Remove Directory)。この操作が成功するかどうかは、ユーザーが持っているリモートホスト上でのユーザー特権に依存します。
SEND file1 file2	file1 に指定されたローカルファイルをリモートホストにコピーして、file2 に指定された名前のリモートファイルを作成します。PUT と同じです。
SIZE file1	file1 に指定されたディレクトリのサイズを返します。

まとめ

インターネットトランスファコントロールを使うと、特にプロトコルを設定することもなく、自動的に必要なプロトコルが実行され、そのままコマンドを使用できます。コマンドの使い方もいたって簡単ですから、容易にFTPやHTTPプロトコルでサーバと通信ができます。

このプログラムでは、「DIR」コマンドしか使いませんでしたが、コマンドをうまく組み合わせれば、フリーソフトでよくあるようなもっと高機能なものや、自分の用途に特化したFTPクライアントソフトも作成できるでしょう。

練習問題

Q1　プログラムにプログレスバーを追加し、ダウンロード中であることを表示してください。

解答は巻末に→

ワンポイントアドバイス

ここでは、匿名（anonymous）によるFTPサーバへのログインを行いましたが、アカウントとパスワードが必要なプライベートサーバにログインする場合は、インターネットトランスファコントロールの次のプロパティを設定してから、Executeメソッドを実行してください。

プロパティ	内容	値
UserName	ユーザー名	文字列
Password	パスワード	文字列

Lesson 20

第６日
３時限目

インターネットプログラムの作成

Lesson20.vbp
VB6CD2
Lesson20

リストビューコントロールを使う

３時限目のレッスンは、次のレッスンで作成するFTPクライアントプログラムに使用する、リストビューコントロールの使い方を覚えます。リストビューコントロールは、複数の項目をリスト表示するコントロールですが、Windowsのフォルダのように、４つの表示形態を持っています。また、リスト項目にアイコンを組み込むことができるので、ファイルの一覧表示などではとても役に立つ便利なコントロールです。しかし、多機能な分設定が少し複雑なので、このレッスンでリストビューコントロールの使い方をマスターして、他のアプリケーションに応用できるようにします。

今回作成するプログラム

アイコン表示（標準）

小さいアイコン表示

リストビュー表示

Report表示

POINT

コードを使用して、リストビューコントロールに50個の項目を設定します。それぞれのモードに合わせて、大きいアイコンと小さいアイコンをリスト項目に組み込みます。また、ツールバーコントロールで、４つの表示形態を切り替えられるようにします。リストビューコントロール内のアイコンをクリックすると、ラベルコントロールにその項目名が表示されます。

コントロールとプロパティの設定

※新しいプロジェクトファイル（標準EXE）を用意します。

1 ツールパレットに「Microsoft Windows Common Controls 6.0」コンポーネントを追加する

ツールパレットにコンポーネントが追加される

2 フォームを広げ、ツールパレットの［ImageList］を使って、イメージリストコントロールを配置する

Form1

1 斜めにドラッグして配置

3 ［プロパティページ］の［イメージ］タブで、［ピクチャの挿入］を使ってイメージを追加する

ファイル名	インデックス	キー
48.ICO	1	
50.ICO	2	
52.ICO	3	
54.ICO	4	
246.ICO	5	flag
SPOT.ICO	6	spot

コンポーネントを追加する

コンポーネントを追加する方法については、「Lesson5 ツールバーコントロールの組み込み」を参照してください。

［プロパティページ］ダイアログ

［プロパティページ］ダイアログボックスを表示するには、コントロールを選択し、プロパティウィンドウの［(プロパティページ)］をクリックし、右欄の［...］ボタンをクリックします。

ピクチャの挿入

［イメージ］タブをクリックし、［ピクチャの挿入］ボタンをクリックします。［図の選択］ダイアログボックスが表示されるので、挿入するピクチャを選択し、［開く］ボタンをクリックします。

ツールパレットの［Toolbar］を使って、ツールバーコントロールを配置する

追加したイメージ

4 ツールパレットの［Toolbar］ を使って、ツールバーコントロールを配置する

1 斜めにドラッグして配置

5 ［プロパティページ］で、［イメージリスト］と［境界線］を設定する

1 ここを「ImageList1」に設定

2 ここを「1－cdFixedSingle」に設定

6 ［ボタン］タブをクリックし、［ボタンの挿入］をクリックしてボタンを４つ組み込む

インデックス	イメージ	キー
1	1	Icon
2	2	SmallIcon
3	3	List
4	4	Report

［プロパティページ］ダイアログ

［プロパティページ］ダイアログボックスを表示するには、コントロールを選択し、プロパティウィンドウの［(プロパティページ)］をクリックし、右欄の［...］ボタンをクリックします。

ボタンを追加する方法

ボタンを追加する方法の詳細については、「Lesson 5 ツールバーコントロールの組み込み」を参照してください。

7 ツールパレットの［ListView］を使って、リストビューコントロールを配置する

1 斜めにドラッグして配置

8 ［プロパティページ］の［イメージリスト］タブで、イメージリストコントロールを設定する

1 「ImageList1」を設定

［プロパティページ］ダイアログ

［プロパティページ］ダイアログボックスを表示するには、コントロールを選択し、プロパティウィンドウの［(プロパティページ)］をクリックし、右欄の［...］ボタンをクリックします。

ラベルコントロールを配置する

ラベルコントロールを配置するには、ツールパレットの［Label］ A を使います。

9 ラベルコントロールを配置し、[BorderStyle] プロパティを「1 ー実線」に設定する

Form1

ListView1

Label1

完成したユーザーインターフェイス

コードの記述

Form1:Form-Load

Lesson20 - Form1 (ｺｰﾄﾞ)

Form　Load

```
Private Sub Form_Load()
    Dim Litem As Object
    Dim ColHead As Object

    ListView1.View = lvwIcon

    For i = 1 To 50
        Set Litem = ListView1.ListItems.Add
            With Litem
            .SmallIcon = "spot"
            .Icon = "flag"
            .Text = "No " & i & "Position"
        End With
    Next
```

```
    For i = 1 To 4
        Set ColHead = ListView1.ColumnHeaders.Add()
        ColHead.Text = "Field " & i
        ColHead.Width = ListView1.Width / 5
    Next

    With ListView1.ListItems(1)
        .SubItems(1) = "Hello"
        .SubItems(2) = "Boy!"
        .SubItems(3) = "I am spot"
    End With
End Sub
```

Form1:ListView1-ItemClick

Lesson20 - Form1 (コード)

ListView1 | ItemClick

```
Private Sub ListView1_ItemClick(ByVal Item As _
                                MSComctlLib.ListItem)
    Label1.Caption = Item.Text
End Sub
```

Form1:Toolbar1-ButtonClick

Lesson20 - Form1 (コード)

Toolbar1 | ButtonClick

```
Private Sub Toolbar1_ButtonClick(ByVal Button As _
    MSComctlLib.Button)
    Dim ViewStyle As Integer

    Select Case Button.Key
        Case Is = "Icon"
            ViewStyle = lvwIcon

        Case Is = "SmallIcon"
            ViewStyle = lvwSmallIcon

        Case Is = "List"
            ViewStyle = lvwList

        Case Is = "Report"
            ViewStyle = lvwReport
    End Select

    ListView1.View = ViewStyle
End Sub
```

解説

1 リストビューコントロールは、「Icon」「SmallIcon」「List」「Report」という、4つの表示形態で項目を表示するコントロールです。これらは、リストビューコントロールの［View］プロパティの値を変えることで、表示形態を切り替えます。

リストビューコントロールに設定できる項目は2つあります。1つは文字列（テキスト）で、もう1つはアイコンです。また、アイコンは表示形態が「Icon」のときは大きいアイコン（32×32）で表示され、それ以外では小さいアイコン（16×16）で表示されます。

リストビューコントロールの項目は、ListItemオブジェクトとして扱われます。したがって、リストビューコントロールに項目を設定するには、このオブジェクトをAddメソッドで追加します。また、テキストやアイコンは、ListItemオブジェクトのプロパティを操作して設定します。

2 このようなコントロールの特徴を踏まえ、フォームのLoadイベントプロシージャで、リストビューコントロールの初期設定を行います。

まず、リストビューコントロールの［View］プロパティで、プログラム起動時の表示形態を「Icon」に設定します。［View］プロパティは、4つの定数を値に持ち、使用したい表示形態をこの定数で設定します。このプログラムでは、標準の大きいアイコンで表示するため、「lvwIcon」を値として設定します。

```
Private Sub Form_Load()
    Dim Litem As Object
    Dim ColHead As Object

    ListView1.View = lvwIcon
```

＜［View］プロパティの値＞

定数	値	内容
lvwIcon	0	（既定値）大きいアイコンで表示。各ListItemオブジェクトは、アイコンおよびテキストラベルで表示されます。
lvwSmallIcon	1	小さいアイコンで表示。各ListItemオブジェクトは、小さいアイコンおよびアイコンの右側のテキストラベルで表示されます。項目は左右に並んで表示されます。
lvwList	2	一覧表示。各ListItemオブジェクトは、小さいアイコンおよびアイコンの右側のテキストラベルで表示されます。各ListItemオブジェクトは上下に並んで、各行に配置され、情報が列方向に表示されます。
lvwReport	3	詳細表示。各ListItemオブジェクトは、小さいアイコンおよびテキストラベルで表示されます。各ListItemオブジェクトの追加情報を、サブアイテムに設定できます。アイコン、テキストラベル、および情報が列方向に表示され、左端の列には小さいアイコンに続けてテキストラベルが表示されます。追加した列には、項目の各サブアイテムのテキストが表示されます。

3 次に、リストビュー内に項目を設定します。ここでは、For...Nextループを使って、一気に50個の項目を組み込みます。リストビューコントロールに項目を追加するには、「ListItems」コレクションオブジェクトを対象として、Addメソッドを実行します。

```
For i = 1 To 50
    Set Litem = ListView1.ListItems.Add
```

追加した「ListItems」コレクションオブジェクトは、Setステートメントで「Litem」というオブジェクト変数に格納しておきます。

次に、このオブジェクト名を使って、アイコンの設定を行います。[Icon] プロパティは大きいアイコンを設定し、[SmallIcon] プロパティは小さいアイコンを設定するプロパティです。どちらも、直接アイコンファイルを指定するのではなく、リストビューコントロールに設定したイメージリストコントロールに組み込まれている、ピクチャのインデックス番号またはキーの値を設定します。

4つのビューで同じアイコンを使いたい場合は、どちらも同じピクチャを設定しますが、ここでは、大小それぞれ別々のピクチャを、「キー」を使って指定しています。

```
With Litem
    .SmallIcon = "spot"
    .Icon = "flag"
    .Text = "No " & i & "Position"
End With
```

項目名のテキストは、[Text] プロパティに文字列を設定します。ここでは、「No1Position」というように、項目順に連番で文字列を作成して設定しています。

Form1

No 1Position　No 2Position　No 3Position　No 4Position　No 5Position
No 6Position　No 7Position　No 8Position　No 9Position　No 10Position
No 11Position　No 12Position　No 13Position　No 14Position　No 15Position

No 7Position

設定した大きいアイコンとテキスト

4 項目のテキストとアイコンの設定ができたら、今度はReport表示時にビューの最上部に設定するサブアイテムの見出しを組み込みます。Reportビューのときだけ、項目名はいくつかの情報を表示させることができます。この情報のことを「サブアイテム」といいます。そして、サブアイテムの見出しになる部分を「コラムヘッダ」といいます。画面の「Field1」「Field2」というのがコラムヘッダで、これはReport表示のみに表示されます。

コラムヘッダ

Reportビュー。項目名の次にサブアイテムが並び、サブアイテムの見出しをコラムヘッダで表示している

サブアイテム

コラムヘッダを組み込むには、Addメソッドでリストビューコントロールに「ColumnHeaders」コレクションオブジェクトを追加します。コラムヘッダはいくつでも追加挿入できますが、必ず項目名から追加されますので、次に作成するサブアイテムよりも1つ多く追加しておく必要があります。

「ColumnHeaders」コレクションオブジェクトを追加したら、そのオブジェクト名をSetステートメントでオブジェクト変数「ColHead」に格納しておきます。

```
For i = 1 To 4
    Set ColHead = ListView1.ColumnHeaders.Add()
```

追加した「ColumnHeaders」コレクションオブジェクトに文字列を表示させるには、このオブジェクトの［Text］プロパティを使います。ここでは、組み込んだ「ColumnHeaders」コレクションオブジェクトを順番に番号を付けています。

また、［Width］プロパティを使うと、「ColumnHeaders」コレクションオブジェクトの幅を指定できます。ここでは、リストビューコントロールの幅を5等分した幅に設定しています。

```
ColHead.Text = "Field " & i
ColHead.Width = ListView1.Width / 5
```

5 　コラムヘッダが設定できたら、サブアイテムを組み込みます。サブアイテムは、項目1つ1つに設定でき、「ListItem」オブジェクトの［SubItems］プロパティを使用します。

まず、［ListItems］プロパティで、サブアイテムを設定したい項目を指定します。次に、［SubItems］プロパティに文字列を設定します。サブアイテムは、項目名の次から順番に「SubItems(1)」「SubItems(2)」と指定していきます。

```
With ListView1.ListItems(1)
    .SubItems(1) = "Hello"
    .SubItems(2) = "Boy!"
    .SubItems(3) = "I am spot"
End With
```

このプログラムでは、最初の項目だけにサブアイテムを設定しました。

6 　リストビューコントロールの初期設定が終わったら、今度はリストビューコントロール内の項目が、ユーザーによってクリックされたときの処理を作成します。

ユーザーが項目名をクリックすると、リストビューコントロールには「ItemClick」イベントが発生します。このイベントプロシージャでは引数「Item」に、クリックされた項目のオブジェクト名が格納されます。このオブジェクトの［Text］プロパティを参照すれば、クリックされた項目名が把握できますので、これをラベルコントロールの［Caption］プロパティに代入して表示しています。

```
Private Sub ListView1_ItemClick(ByVal Item As _
                                MSComctlLib.ListItem)
    Label1.Caption = Item.Text
End Sub
```

ラベルコントロールにクリックした項目名が表示される

7 最後に、ツールバーコントロールのボタンがクリックされたときに、リストビューコントロールの表示形態を変更する処理を作成します。

ツールバーコントロールの「ButtonClick」イベントプロシージャで、引数「Button」に格納されたButtonオブジェクトの［Key］プロパティの値を使って、Select Caseステートメントで処理を分岐しています。そして、それぞれのボタンがクリックされたとき、変数「ViewStyle」に［View］プロパティの定数を格納し、［View］プロパティに渡しています。これで、ツールバーのボタンをクリックすれば、リストビューコントロールはそれぞれのビューに切り替えて表示するようになります。

```
Private Sub Toolbar1_ButtonClick(ByVal Button As _
                                MSComctlLib.Button)
    Dim ViewStyle As Integer

    Select Case Button.Key
        Case Is = "Icon"
            ViewStyle = lvwIcon

        Case Is = "SmallIcon"
            ViewStyle = lvwSmallIcon

        Case Is = "List"
            ViewStyle = lvwList

        Case Is = "Report"
            ViewStyle = lvwReport
    End Select

    ListView1.View = ViewStyle
End Sub
```

まとめ

リストビューコントロールは４つの違う表示形態を持っているため、プロパティの操作が多くなっています。また、項目やサブアイテムなど、いくつかのオブジェクト（またはコレクションオブジェクト）で構成されていますので、どの部分を変更するときはどのオブジェクトのプロパティを使うのかをよく把握しておく必要があります。

練習問題

Q1　リストビューコントロールの項目名は、その場で変更できるようになっています。そこで、ユーザーが項目名を変更しようとしたら、メッセージボックスを表示し、[OK] ボタンがクリックされれば変更し、[キャンセル] ボタンがクリックされれば変更を取り消すように、プログラムを修正してください。

項目名をプログラム実行時に変更できる

メッセージボックスで変更を確認する

解答は巻末に→

ワンポイントアドバイス

「Icon」および「SmallIcon」のビューでは、ユーザーがマウスでアイコンを移動させることができます。また、項目名の変更はすべてのビューで行えます。

アイコンを移動できる

Lesson 21

第６日
４時限目

インターネットプログラムの作成

Lesson21.vbp
VB6CD2
Lesson21

簡単FTPクライアントプログラムの作成

リストビューコントロールの使い方がわかったところで、Lesson19で作成したFTPプロトコルを操作するプログラムに、このリストビューコントロールを組み込み、サーバのディレクトリ一覧をリストビューコントロールで表示するようにします。また、リストビュー内のアイコンを選択してFTPプロトコルを使い、サーバのファイルを自分のコンピュータに転送する機能も組み込みます。

今回作成するプログラム

サーバーのディレクトリの一覧をアイコン付きで表示し...

自分のコンピュータに転送できるようにする

POINT

インターネットトランスファーコントロールのFTPプロトコル転送で、サーバのディレクトリ一覧を取得し、リストビューコントロールに表示します。リストビューコントロールは、表示形態を「List」表示にして、ファイルとディレクトリで別々のアイコンを組み込みます。また、クリックしたアイコンを、FTPプロトコルでダウンロードできるようにします。

コントロールとプロパティの設定

※「Q&A-19.vbp」のプロジェクトファイルを開いてください。このプログラムは、「Q&A-19.vbp」にコントロールと機能を追加して作成します。

1 ツールパレットの［ImageList］を使って、フォームにイメージリストコントロールを配置する

Mini FTP Cliant
URLの入力: 例 ftp.microsoft.com
ftp.microsoft.com
Get Dir List
Exit
RichText

1 斜めにドラッグして配置

2 ［プロパティページ］の［全般］タブで、「16 ×16」サイズを指定する

ﾌﾟﾛﾊﾟﾃｨ ﾍﾟｰｼﾞ
全般 | ｲﾒｰｼﾞ | 色
16 x 16(1)
32 x 32(3)
48 x 48(4)
ﾕｰｻﾞｰ設定(C)
MaskColor の使用(U)
高さ(H):
幅(W): 16
OK | ｷｬﾝｾﾙ | 適用(A) | ﾍﾙﾌﾟ

1 ここをクリック

Q&A-19.vbpファイル

「Q&A-19.vbp」ファイルは、付録CD-ROMの「Q&A」フォルダにも収録されています。

フォームを拡げる

フォームのサイズを拡げておいてください。

Height：約6300
Width：約6800

［プロパティページ］ダイアログ

［プロパティページ］ダイアログボックスを表示するには、コントロールを選択し、プロパティウィンドウの［(プロパティページ)］をクリックし、右欄の［...］ボタンをクリックします。

ピクチャの挿入

［イメージ］タブをクリックし、［ピクチャの挿入］ボタンをクリックします。［図の選択］ダイアログボックスが表示されるので、挿入するピクチャを選択し、［開く］ボタンをクリックします。

3 ［イメージ］タブをクリックし、［ピクチャの挿入］を使ってアイコンファイルを追加する

ファイル名	インデックス	キー
FILECB09.ICO	1	folder
DOC12.ICO	2	doc

4 ツールパレットの［ListView］を使って、リストビューコントロールを配置する

5 ［プロパティページ］の［イメージリスト］タブで、［小さいアイコン］に「ImageList1」を設定する

6 ツールパレットの［CommandButton］を使ってコマンドボタンを配置し、プロパティを変更する

プロパティ	値
オブジェクト名	DownBtn
Caption	Download

［プロパティページ］ダイアログ

［プロパティページ］ダイアログボックスを表示するには、コントロールを選択し、プロパティウィンドウの［(プロパティページ)］をクリックし、右欄の［...］ボタンをクリックします。

複数項目の選択を可能にする

リストビュー内の項目を複数選択できるようにしたいときは、コントロールのMultiSelectプロパティをTrueに設定します。

［プロパティページ］ダイアログボックスからは［全般］タブで［複数校目の選択］をチェックしておけば自動的にそのリスト内の項目をCtrlキーやShiftキーなどで複数選択できるようになります。

コードの記述

<追加するコード>

Form1:(General)-(Declarations)

Lesson21 - Form1 (ｺｰﾄﾞ)

(General) / (Declarations)

```
Dim GetUrl As String
Dim GetFile As String
Dim Flag As Boolean
Dim i As Integer
```

Form1:(General)-Listup

Lesson21 - Form1 (ｺｰﾄﾞ)

(General) / Listup

```
Private Sub Listup()
    Dim DirItem As String
    Dim MarkStr As String

    Open "c:¥temp.txt" For Input As #1
    Do While Not EOF(1)
        Set Litem = ListView1.ListItems.Add

        Line Input #1, DirItem
        MarkStr = Right(DirItem, 1)
        If DirItem <> "" Then
            If MarkStr = "/" Then
                Litem.SmallIcon = "folder"
            Else
                Litem.SmallIcon = "doc"
            End If
        End If
        Litem.Text = DirItem
    Loop
    Close #1
End Sub
```

Form1:DownBtn-Click

Lesson21 - Form1 (コード)

DownBtn | Click

```
Private Sub DownBtn_Click()
    Flag = False
    Inet1.Execute GetUrl, GetFile
End Sub
```

Form1:Form-Load

Lesson21 - Form1 (コード)

Form | Load

```
Private Sub Form_Load()
    ListView1.View = lvwList
    i = 1
End Sub
```

＜修正したプロシージャ＞

Form1:GetBtn-Click

Lesson21 - Form1 (コード)

GetBtn | Click

```
Private Sub GetBtn_Click()
    If ListView1.ListItems.Count >= 1 Then
        ListView1.ListItems.Clear
    End If
    GetUrl = Combo1.Text
    GetUrl = "ftp://" & GetUrl

    Inet1.Execute GetUrl, "DIR"
    Flag = True
End Sub
```

Form1:Inet1-StateChanged

Lesson21 - Form1 (コード)

Inet1 | StateChanged

```
Private Sub Inet1_StateChanged(ByVal State As Integer)
    Dim GetData As Variant

    Select Case State
```

```
        Case icResponseCompleted
            If Flag = True Then
                GetData = Inet1.GetChunk(2048, icString)
                RichTextBox1.Text = GetData
            End If
            MsgBox "ダウンロード完了"
            ProgressBar1.Value = 0
            i = 0

        Case icReceivingResponse
            i = i + 1
            If i = 10 Then
                i = 0
            End If
            ProgressBar1.Value = i

        Case icError
            MsgBox "ダウンロードできません"
            Inet1.Execute , "CLOSE"
            ProgressBar1.Value = 0
            i = 0
        End Select
End Sub
```

＜変更しないコード＞

Form1:RichTextBox1-Change

Lesson21 - Form1 (ｺｰﾄﾞ)

RichTextBox1 | Change

```
Private Sub RichTextBox1_Change()
    RichTextBox1.SaveFile "c:¥temp.txt", 1
    Call Listup
End Sub
```

Form1:ListView1-ItemClick

Lesson21 - Form1 (ｺｰﾄﾞ)

ListView1 | ItemClick

```
Private Sub ListView1_ItemClick(ByVal Item As _
   MSComctlLib.ListItem)
    GetFile = "GET " & Item.Text & " " & "c:¥" & Item.Text
End Sub
```

解説

1 このプログラムは、Lesson19の練習問題で作成したインターネットトランスファーコントロールのFTPプロトコル機能を使って、FTPサーバのディレクトリを取得するプログラムに、リストビューコントロールとプログレスバーを組み込みます。サーバから取得したディレクトリ一覧は、リッチテキストボックスコントロールに表示されると同時に、リストビューコントロールに、ファイルとディレクトリのアイコンを区別して表示します。

また、サーバから転送中は、プログレスバーコントロールに、転送中の状況を表示します。

アイコン表示されたリストビューコントロールのファイル名をクリックして、［Download］ボタンをクリックすると、ファイルをサーバから自分のコンピュータに転送します。

2 では、コードを作成していきましょう。まず、フォームの宣言セクションに、4つの変数をモジュールレベル変数で宣言します。

「GetUrl」は、コンボボックスコントロールから取得した、FTPサーバのURLを格納します。「GetFile」は、リストビューコントロールでクリックされたファイル名を格納する変数で、文字列で宣言します。変数「Flag」は、ディレクトリ情報の転送なのか、ファイルの転送なのかを区別するために使用します。論理型で宣言し、値が「True」でディレクトリ情報の転送、「False」でファイルの転送にします。変数「i」は、プログレスバーの進行状況を設定するのに使用する変数です。

```
Dim GetUrl As String
Dim GetFile As String
Dim Flag As Boolean
Dim i As Integer
```

フォームのLoadイベントプロシージャで、リストビューコントロールを「List」表示に設定します。このプログラムでは、1つのビューだけを使うようにしています。また、変数「i」を初期化しておきます。

```
Private Sub Form_Load()
    ListView1.View = lvwList
    i = 1
End Sub
```

3 次に、[Get Dir List] ボタンのClickイベントプロシージャを修正します。ここでは、まずリストビューコントロールにリスト項目が設定されているかどうかをチェックします。すでにリスト項目があるのに、Addメソッドで新たに項目を追加してしまうと、複数のFTPサーバのディレクトリ情報が混ざってしまいます。そこで、すでにリスト表示があれば、それを消しておく必要があります。

リストビューコントロールの [Count] プロパティの値を調べると、現在リストビューコントロールに項目の表示があるかどうかがわかります。もしあれば、「Clear」メソッドを使用して項目をすべて消します。

```
Private Sub GetBtn_Click()
    Dim GetUrl As String

    If ListView1.ListItems.Count >= 1 Then
        ListView1.ListItems.Clear
    End If
```

そして、Lesson19で作成したディレクトリ情報の取得を、インターネットトランスファーコントロールのExecuteメソッドで行います。

```
GetUrl = Combo1.Text
GetUrl = "ftp://" & GetUrl

Inet1.Execute GetUrl, "DIR"
```

4 最後に、変数「Flag」に「True」を代入して、ディレクトリ情報の取得を行ったことを記録しておきます。

```
Flag = True
```

［Get Dir List］ボタンでサーバからの転送を開始すると、インターネットトランスファーコントロールの「StateChanged」イベントが発生します。引数「State」の値をSelect Caseステートメントで調べます。

転送が完了した「icResponseCompleted」が格納されていれば、しかもそれがディレクトリ情報の取得であれば（Flag = Trueであれば）、「GetChunk」メソッドを実行して、リッチテキストボックスコントロールにディレクトリの内容を転送します。そして、ダウンロード完了のメッセージボックスを表示し、プログレスバーの表示と変数「i」をリセットします。

```
Private Sub Inet1_StateChanged(ByVal State As Integer)
    Dim GetData As Variant

    Select Case State
        Case icResponseCompleted
            If Flag = True Then
                GetData = Inet1.GetChunk(2048, icString)
                RichTextBox1.Text = GetData
            End If
            MsgBox "ダウンロード完了"
            ProgressBar1.Value = 0
            i = 0
```

サーバからの転送中は、引数「State」には「icReceivingResponse」が格納されているので、このときにプログレスバーコントロールの［Value］プロパティに値を設定して、転送中の表示を行います（Q&A-19を参照）。

```
Case icReceivingResponse
    i = i + 1
    If i = 10 Then
        i = 0
    End If
    ProgressBar1.Value = i
```

また、転送中にエラーが発生した場合は、「State」には「icError」が格納されるので、そのときはエラーのメッセージボックスを表示し、FTPサーバとの接続を切断します。そして、プログレスバーの表示と変数「i」をリセットし、プロシージャを終了します。

```
Case icError
    MsgBox "ダウンロードできません"
    Inet1.Execute , "CLOSE"
    ProgressBar1.Value = 0
    i = 0
```

5 リッチテキストボックスコントロールに、ディレクトリ情報が格納されると、リッチテキストボックスコントロールには「Change」イベントが発生します。そこで、このイベントプロシージャで、リッチテキストボックスコントロールの内容をテンポラリファイル「c:¥temp.txt」に格納します。同時に、リッチテキストボックスコントロールの内容を使って、リストビューコントロールの項目表示を実行します。これは、ユーザー定義のプロシージャ「Listup」に作成し、これを呼び出します。

```
Private Sub RichTextBox1_Change()
    RichTextBox1.SaveFile "c:¥temp.txt", 1
    Call Listup
End Sub
```

6 リストビューコントロールの設定を行うプロシージャ「Listup」を、新しく作成します。このプロシージャは、テンポラリファイルに格納してあるディレクトリ情報を1行ずつ解析し、項目名を作成しアイコンといっしょにリストビューコントロールに追加します。アイコンの設定は、項目名がディレクトリであれば、ディレクトリ用のアイコンを、そうでない場合はファイル用のアイコンを組み込み、アイコンで区別が付けられるようにします。

ファイルとディレクトリを項目名に設定し、アイコンを変えて表示する

まず、変数を3つ宣言します。変数「DirItem」は、ディレクトリ情報を格納しているテンポラリファイルから取り出した、ディレクトリまたはファイル名を格納します。また、変数「MarkStr」は、取り出したディレクトリ名の末尾に「/」(ディレクトリであることの記号)が付いているかどうかを判別するのに使用します。

```
Private Sub Listup()
    Dim DirItem As String
    Dim MarkStr As String
```

続いて、Openメソッドで、テンポラリファイルを開きます。

```
Open "c:¥temp.txt" For Input As #1
```

そして、Do...While Loopステートメントで、ファイルの終端にくるまでループをまわします。ループの中では、Addメソッドを使ってリストビューコントロールに項目を設定します。追加した項目は、Setステートメントでオブジェクト変数「Litem」に、オブジェクト名を格納します。

```
Do While Not EOF(1)
    Set Litem = ListView1.ListItems.Add
```

7

次に、「Line Input #」ステートメントで、ファイルの中身を1行だけ取り出します。取り出した1行は、「Right」関数で、末尾の1文字を取り出し、変数「MarkStr」に格納します。

```
Line Input #1, DirItem
MarkStr = Right(DirItem, 1)
```

Ifステートメントを使って、変数「DirItem」が空でないかどうかをチェックします。通信の状態で""のデータが転送される場合があるからです。

変数の中身がディレクトリ名またはファイル名であれば、今度は、末尾の文字が「/」でないかどうかを調べます。変数「MarkStr」の値が「/」であれば、それはディレクトリ名なので、項目のアイコンを表示する[SmallIcon]プロパティに「folder」を設定します。この名前は、イメージリストコントロールに追加したフォルダ用のアイコンのキーの値です。「/」でなければそれはファイルなので、[SmallIcon]プロパティにキー「doc」を設定します。

プロパティページ
全般 イメージ 色
現在のイメージ
インデックス(I): 1 キー(K): folder
タグ(T):
イメージ(M):
ピクチャの挿入(P)... ピクチャの削除(R) イメージ数: 2
OK キャンセル 適用(A) ヘルプ

イメージリストコントロールのキーを［SmallIcon］プロパティに設定する

```
If DirItem <> "" Then
    If MarkStr = "/" Then
    Litem.SmallIcon = "folder"
    Else
    Litem.SmallIcon = "doc"
    End If
End If
```

アイコンを設定したら、今度は［Text］プロパティにディレクトリ名またはファイル名を設定します。これらの処理は、ディレクトリ情報が格納されているファイルの終端まで行われます。そして、サーバから取得したディレクトリ情報をすべて、リッチテキストボックスコントロールとリストビューコントロールに表示します。

```
        Litem.Text = DirItem
    Loop
    Close #1
End Sub
```

そして、「Close」ステートメントで、開いていたファイルを閉じます。

8 今度は、リストビューコントロールに表示されたファイルをダウンロードする処理を作成しましょう。

リストビューコントロールの項目がユーザーによってクリックされると、「ItemClick」イベントが発生します。このイベントプロシージャは引数「Item」を持ち、クリックされた「ListItem」オブジェクトを返してきます。そこで、この「ListItem」オブジェクトの［Text］プロパティから、クリックされた項目名を取得します。

インターネットトランスファーコントロールのExecuteメソッドは、FTPプロトコルを使う場合は、「GET」コマンドを実行すると、サーバにあるファイルを接続しているコンピュータに転送できます。この場合は、「GET」コマンドのあとに、転送するファイル名と転送先のドライブ・ファイル名を指定します。そこで、取得した項目名を使って、転送するファイル名と、転送先のファイル名をフルパスで作

成し、「GET」コマンドを実行します。ここではCドライブのルートディレクトリにファイルを転送します。

そこで、このプロシージャでは、「GET」コマンドの記述を作成して、モジュールレベルの変数「GetFile」に格納しています。

```
Private Sub ListView1_ItemClick(ByVal Item As _
    MSComctlLib.ListItem)
    GetFile = "GET " & Item.Text & " " & "c:¥" & Item.Text
End Sub
```

一方、［DownLoad］ボタンがクリックされると、ダウンロード操作がディレクトリの取得ではなくファイルの取得なので、変数「Flag」に「False」を設定し、「Execute」コマンドを実行します。

```
Private Sub DownBtn_Click()
    Flag = False
    Inet1.Execute GetUrl, GetFile
End Sub
```

これで、リストビューコントロールでクリックしたファイルは、コンピュータに転送されます。転送中は、インターネットトランスファーコントロールの「StateChanged」イベントが発生します。

変数「Flag」を「False」に設定しているので、リッチテキストボックスコントロールへのディレクトリ表示は実行されず、転送中をあらわすプログレスバーの表示が実行され、ダウンロードが終了するとメッセージボックスが表示されます（プロシージャ「Inet1_StateChanged(ByVal State As Integer)」を参照）。

! ワンポイントアドバイス

すでにダウンロードしたファイルがコンピュータに存在するにも関わらず、もう一度ダウンロードしようとすると、インターネットトランスファーコントロールはエラーを起こします。エラー処理の定数は次ページに表にしているので、これを参考にしながら、必要ならエラー処理を組み込んでおきましょう。

また、このプログラムはサーバのルートディレクトリの情報を取得していますが、サーバ内の各ディレクトリの情報を取得するには、「CD」コマンドなどを使用します。Executeメソッドで実行できるFTPコマンドについてはLesson19を参照してください。

まとめ

以上で、プログラムが完成です。インターネットに接続して、FTPサーバを指定して、ディレクトリの取得とファイルのダウンロードを実行してください。

インターネットトランスファーコントロールのFTPプロトコルは、まだまだたくさんのコマンドを用意していますから、フリーソフトや市販のFTPクライアントソフトのように、ディレクトリの作成や変更、ドラッグ＆ドロップの採用などいろいろな機能を組み込んだソフトに仕上げることができます。

練習問題

Q1 このプログラムでは、ファイルの転送先を「C:¥」に固定して転送しました。これを、コモンダイアログボックスを使って、ファイル名を変えずに好きなフォルダを指定してダウンロードできるように修正してください。

解答は巻末に→

ワンポイントアドバイス

このプログラムはサンプルなので、充分なエラー処理を組み込んでいません。実際にソフトを開発する場合は、以下の表を参考にしてエラー処理を組み立ててください。

<インターネットトランスファコントロールのトラップできるエラーおよびその定数の一覧>

定数	値	内容
icOutOfMemory	7	メモリが不足しています。
icTypeMismatch	13	型が一致しません。
icInvalidPropertyValue	380	プロパティの値が不正です。
icInetOpenFailed	35750	インターネットハンドルを開けません。
icOpenFailed	35751	URLを開けません。
icBadUrlL	35752	URLが不正です。
icProtMismatch	35753	このメソッドでサポートされていないプロトコルです。
icConnectFailed	35754	リモートホストに接続できません。
icNoRemoteHost	35755	リモートコンピュータが指定されていません。
icRequestFailed	35756	要求を完了できません。
icNoExecute	35757	データを抽出する前に、任意の操作を実行しておく必要があります。

定数	値	内容
icBlewChunk	35758	データを抽出できません。
icFtpCommandFailed	35759	FTPコマンドに失敗しました。
icUnsupportedType	35760	タイプは強制的には指定できません。
icTimeOut	35761	要求がタイムアウトになりました。
icUnsupportedCommand	35762	不正なコマンド、あるいはサポートされないコマンドです。
icInvalidOperation	35763	操作の引数が不正です。
icExecuting	35764	最後の要求が実行中です。
icInvalidForFtp	35765	この呼び出しはFTP接続の場合には不正です。
icOutOfHandles	35767	ハンドル外です。
icInetTimeout	35768	タイムアウトです。
icExtendedError	35769	拡張エラーです。
icIntervalError	35770	内部エラーです。
icInvalidURL	35771	不正なURLです。
icUnrecognizedScheme	35772	認識されないスキームです。
icNameNotResolved	35773	名前を特定できません。
icProtocolNotFound	35774	プロトコルが見つかりません。
icInvalidOption	35775	不正なオプションです。
icBadOptionLength	35776	オプションの長さに問題があります。
icOptionNotSettable	35777	オプションを設定できません。
icShutDown	35778	シャットダウンします。
icIncorrectUserName	35779	ユーザー名が正しくありません。
icLoginFailure	35781	ログオンに失敗しました。
icInetIvalidOpertation	35782	不正な操作です。
icOperationCancelled	35783	操作がキャンセルされました。
icIncorrectHandleType	35784	ハンドルの種類が正しくありません。
icIncorrectHandleState	35785	ハンドルの状態が正しくありません。
icNotProxyRequest	35786	プロキシではありません。
icRegistryValueNotFound	35787	レジストリの値が見つかりません。
icbadRegistryParameter	35788	レジストリのパラメータに問題があります。
icNoDirectAccess	35789	ダイレクトアクセスはありません。
icIncorrectPassword	35779	パスワードが正しくありません。
icNoContext	35790	コンテクストがありません。
icNoCallback	35791	コールバックがありません。
icRequestPending	35792	要求は保留されました。
icIncorrectFormat	35793	形式が正しくありません。
icItemNotFound	35794	アイテムが見つかりません。
icCannotConnect	35795	接続できません。

定数	値	内容
icConnectionAborted	35796	接続が中断されました。
icConnectionReset	35797	接続がリセットされました。
icForceRetry	35798	再試行します。
icInvalidProxyRequest	35799	不正なプロキシ要求です。
icWouldBlock	35800	ブロック化を行います。
icHandleExists	35802	ハンドルが存在します。
icSecCertDateInvalid	35803	セキュリティ保証の日付が無効です。
icSecCertCnInvalid	35804	セキュリティ保証の数値が不正です。
icHttpsToHttpOnRedir	35806	HTTPS から HTTP へのリダイレクトです。
icMixedSecurity	35807	セキュリティが混在しています。
icChgPostIsNonSecure	35808	投稿の変更はセキュリティ上問題がある可能性があります。
icHttpToHttpsOnRedir	35805	HTTP から HTTPS へのリダイレクトです。
icPostIsNonSecure	35809	投稿はセキュリティ上問題がある可能性があります。
icClientAuthCertNeeded	35810	クライアントを認証する必要があります。
icInvalidCa	35811	クライアントの認証が不正です。
icClientAuthNotSetup	35812	クライアントの認証が設定アップされていません。
icAsyncThreadFailed	35813	非同期スレッドに失敗しました。
icRedirectSchemeChange	35814	スキームの変更をリダイレクトします。
icFtpTransferInProgress	35876	FTP - 転送中です。
icFtpDropped	35877	FTP - 接続に失敗しました。
icGopherProtocolError	35896	Gopher - プロトコルエラーです。
icGopherNotFile	35897	Gopher - ファイルではありません。
icGopherDataError	35898	Gopher - データ エラーが発生しました。
icGopherEndOfData	35899	Gopher - これ以上データはありません。
icGopherInvalidLocator	35900	Gopher - 不正なロケータです。
icGopherIncorrectLocatorType	35901	Gopher - ロケータの種類が正しくありません。
icGopherNotGopherPlus	35902	Gopher - Gopher Plus ではありません。
icGopherAttributeNotFound	35903	Gopher - 属性が見つかりません。
icGopherUnknownLocator	35904	Gopher - 不明なロケータです。
icHeaderNotFound	35916	HTTP - ヘッダーが見つかりません。
icHttpDownlevelServer	35917	HTTP - ダウンレベルのサーバーです。
icHttpInvalidServerResponse	35918	HTTP - サーバーの応答が不正です。
icHttpInvalidHeader	35919	HTTP - 不正なヘッダです。
icHttpInvalidQueryRequest	35920	HTTP - 不正なクエリー要求です。
icHttpHeaderAlreadyExists	35921	HTTP - ヘッダが既に存在します。
icHttpRedirectFailed	35922	HTTP - リダイレクトに失敗しました。
icSecurityChannelError	35923	セキュリティチャネルエラーです。
icUnableToCacheFile	35924	キャッシュファイルが使用できません。

V i s u a l B a s i c 6

Let's VBSchool!!

第7日 (Lesson22～Lesson24) カスタムWebブラウザの作成

L e a r n i n g S c h o o l

コンピュータに、MicrosoftのWWWブラウザ「インターネットエクスプローラ」をインストールしていると、このオブジェクトのインスタンスをVisual Basicで作成して操作できます。また、インターネットエクスプローラを構成しているオブジェクトの1つに、「WebBrowser」コントロールがあります。インターネットエクスプローラ自身のブラウザ機能は、このコントロールをベースに組み立てられており、ActiveXコントロールとしてコンピュータにインストールされています。そのため、この「WebBrowser」コントロールをフォームに組み込むことで、独自のブラウザを作成することができます。

今日のレッスンでは、Visual Basicで作成したフォームから、インターネットエクスプローラを起動して操作するプログラムの作成と、WebBrowserコントロールをフォームに組み込んで、オリジナルのWebブラウザを作成します。

今日のレッスンで解説する項目

- ❑ CreateObject関数によるActiveXオブジェクトのインスタンスの作成
- ❑ URLの設定とページの表示
- ❑ インターネットエクスプローラの表示を設定するプロパティ
- ❑ Internet Client SDKについて
- ❑ Webナビゲーション用メソッドの使い方
- ❑ WebBrowserコントロールの機能と特徴
- ❑ WebBrowserコントロールのプロパティを使う
- ❑ フォーム上の他のコントロールとの連携操作
- ❑ WebBrowserコントロールのイベントを使った処理
- ❑ 転送バイト数の把握
- ❑ 表示しているHTMLドキュメントの保存方法

Lesson 22

第７日
１時限目

カスタム
Webブラウザ
の作成

Lesson22.vbp
VB6CD2
Lesson22

インターネットエクスプローラを操作する

コンピュータに、インターネットエクスプローラのバージョン「3.0x」または「4.0x」以降が組み込まれていれば、これらをVisual Basicのプログラムから、リモートコントロールで操作することができます。そこで、このレッスンでは、プログラムからインターネットエクスプローラのインスタンスを作成して、コンボボックスコントロールに設定したHTMLファイルやホームページを表示するプログラムを作成します。

今回作成するプログラム

このブラウザは、Visual Basicから起動しました。

新しいブラウザの起動
ブラウザの表示切り替え
ブラウザを閉じる

ボタンをクリックするだけで...

インターネットエクスプローラを起動できる

POINT

フォームに配置したコンボボックスコントロールから、HTMLファイル名やURLを選び、「新しいブラウザ」ボタンを押すと、インターネットエクスプローラのインスタンスを生成して、HTMLコンテンツを表示します。プログラムは、インターネットエクスプローラのインスタンスを毎回生成してHTMLを表示する方法と、１つのインスタンスで表示するHTMLを切り替える方法の、２通りのテクニックを使用します。

※この日のレッスンでは、操作するWWWブラウザとしてインターネットエクスプローラ5.0を使用したプログラム作成を解説しています（メソッドの変更で3.xxにも応用できます）。インターネットエクスプローラ2.xxおよびNetscape Navigatorは、操作することができません。

コントロールとプロパティの設定

※新しいプロジェクトファイル（標準EXE）を用意します。また、「material」フォルダ内の「ie.htm」「ie2.htm」「alert.htm」をCドライブにコピーしておいてください。

1 フォームの［Caption］プロパティを「Internet Explorer Control Pad」に変更する

2 ツールパレットの［ComboBox］を使って、コンボボックスコントロールを配置する

3 ［List］プロパティを次のように設定する

c:¥ie.htm
c:¥ie2.htm
c:¥alert.htm
http://www.big.or.jp/~seto/

4 [Text] プロパティを「c:¥ie.htm」に変更する

プロパティ - Combo1

Combo1 ComboBox

全体 | 項目別

TabIndex	3
TabStop	True
Tag	
Text	c:¥ie.htm
ToolTipText	
Top	360
Visible	True
WhatsThisHelpID	0
Width	2325

1 ここを変更

コマンドボタンを配置する

コマンドボタンを配置するには、ツールパレットの［CommandButton］を使います。

5 フォーム上にコマンドボタンを３つ配置し、各ボタンのプロパティを次のように設定する

ボタン	オブジェクト名	Caption	Enabled
Command1	RunBtn	新しいブラウザの起動	True
Command2	ViewBtn	ブラウザの表示切り替え	False
Command3	CloseBtn	ブラウザを閉じる	False

Internet Explorer Control Pad

c:¥ie.htm

新しいブラウザの起動

ブラウザの表示切り替え

ブラウザを閉じる

配置してプロパティを設定したコマンドボタン

6 ［新しいブラウザの起動］ボタンの［TabIndex］プロパティを「0」に変更する

プロパティ - RunBtn	
RunBtn CommandButton	
Picture	(なし)
RightToLeft	False
Style	0 - 標準
TabIndex	0
TabStop	True
Tag	
ToolTipText	
Top	360
UseMaskColor	False
Visible	True

1 「0」に変更

7 完成したフォーム

Internet Explorer Control Pad

c:¥ie.htm
c:¥ie.htm
c:¥ie2.htm
c:¥alert.htm
http://www.big.or.jp/~seto/

新しいブラウザの起動
ブラウザの表示切り替え
ブラウザを閉じる

［TabIndex］プロパティ

フォームに配置したコントロールの［TabIndex］プロパティを「0」に設定すると、フォーム表示時にそのコントロールがフォーカスを持ちます。プログラム起動時に、最初に操作するコントロールに対し、この［TabIndex］プロパティを「0」に設定しておけば、ユーザーはすぐにそのコントロールを操作できるようになります。

コードの記述

Form1:(General)-(Declarations)

Lesson22 - Form1 (ｺｰﾄﾞ)

(General) | (Declarations)

```
Dim Ie As Object
```

Form1:RunBtn-Click

Lesson22 - Form1 (ｺｰﾄﾞ)

RunBtn | Click

```
Private Sub RunBtn_Click()
    Set Ie = CreateObject("InternetExplorer.application")

    Ie.Navigate2 Combo1.Text
    Ie.Visible = True
    Ie.StatusText = "Hello!"
    ViewBtn.Enabled = True
    CloseBtn.Enabled =True
End Sub
```

Form1:ViewBtn-Click

Lesson22 - Form1 (ｺｰﾄﾞ)

ViewBtn | Click

```
Private Sub ViewBtn_Click()
    Ie.Navigate2 Combo1.Text
End Sub
```

Form1:CloseBtn-Click

Lesson22 - Form1 (ｺｰﾄﾞ)

CloseBtn | Click

```
Private Sub CloseBtn_Click()
    Ie.Visible = False
End Sub
```

解　説

1　Visual Basicは、ActiveXテクノロジーで作成されているアプリケーションであれば、プログラムからそのインスタンスを作成して、そのアプリケーションが持つオブジェクトやプロパティ、メソッドなどを操作することができます。インターネットエクスプローラも、ActiveXを使って作成されたアプリケーションなので、「ActiveXオブジェクト」としてVisual Basicのプログラムから、自由に操作することができます。

このレッスンでは、インターネットエクスプローラのプロパティを操作し、リモートコントロールでHTMLを表示させるプログラムを作成していきます。では、簡単にプログラムの動作を説明します。

①コンボボックスコントロールのリストボックスから、表示したいHTMLのURLを選びます。もちろん、コンボボックスですから、テキスト入力エリアにHTMLのURLを直接入力してもかまいません。

②［新しいブラウザの起動］ボタンをクリックすると、インターネットエクスプローラのインスタンスを生成し、コンボボックスコントロールで選択されているURLを表示します。インターネットエクスプローラのインスタンスは、このボタンをクリックするたびにいくつも起動できます。

ボタンをクリックするだけで...

複数のインターネットエクスプローラを起動できる

③［ブラウザの表示切り替え］ボタンをクリックすると、最後に生成したインターネットエクスプローラの表示を、コンボボックスコントロールで選択したURLに入れ替えます。

④［ブラウザを閉じる］ボタンをクリックすると、最後に生成したインターネットエクスプローラのインスタンスを非表示にします。

ワンポイントアドバイス

インターネットエクスプローラのオブジェクトと付随するプロパティ、メソッドはVisual Basicのヘルプやマニュアルには記述されていません。これらは、Microsoftが提供している「Internet Client SDK」という開発キットの中に、詳しく記述されたドキュメントが入っています（詳しくは、このレッスン末尾のコラム「Internet Client SDK」を参照）。

2 他のアプリケーションのインスタンスを作成するには、「CreateObject」関数を使用します。この関数は、指定したオブジェクトのインスタンスを生成し、そのオブジェクト名を返してきます。

この関数の引数に、インターネットエクスプローラの「プログラムID（クラス）」を設定して実行すれば、インターネットエクスプローラのインスタンスが作成されます。「プログラムID」は、そのオブジェクトのクラスを文字列で表したもので、それぞれのアプリケーションごとに独自の名前が付けられています。このクラスがアプリケーションであれば、「Application」クラスに属しますので、クラスとアプリケーションの名前が「プログラムID」になります。

CreateObject関数の書式

```
CreateObject(class)
```

class　引数**class**は「**appname.objecttype**」で指定します。

- **appname**　オブジェクトを提供しているアプリケーションの名前を指定します。
- **objecttype**　作成するオブジェクトの種類またはクラスを指定します。

<代表的なプログラムID>

プログラム名	プログラムID
Microsoft Excel	Excel.application
Microsoft Word	Word.application
インターネットエクスプローラ	InternetExplorer.application

「CreateObject」関数の引数に、この「プログラムID」を設定して実行すると、オブジェクトのインスタンスを作成して、オブジェクト名を戻り値として返してき

ます。返ってきた戻り値は、Setステートメントでオブジェクト変数に格納します。これで、インスタンスのプロパティやメソッドにアクセスできるようになります。

3　では、コードを作成しましょう。

まず、フォームの宣言セクションに、インスタンスのオブジェクト名を格納するオブジェクト変数「Ie」を宣言します。オブジェクト変数なので、データ型は「Object」にします。データ型を「Object」にしておくと、プログラム実行時に「CreateObject」関数が戻してくるオブジェクトの型に設定されます。

```
Dim Ie As Object
```

次に、［新しいブラウザの起動］ボタンのClickイベントプロシージャに、インターネットエクスプローラのインスタンスを作成する処理を記述します。まず、「CreateObject」関数を実行して、インスタンスを作成します。引数には、「InternetExplorer.application」を文字列で記述します。関数の戻り値は、Setステートメントでオブジェクト変数「Ie」に格納します。

```
Set Ie = CreateObject("InternetExplorer.application")
```

これで、インターネットエクスプローラのインスタンスが作成できました。ただし、これだけではインスタンスを作成したにすぎませんので、表示するにはこのオブジェクトのプロパティを操作します。

4　インターネットエクスプローラオブジェクトの「Navigate2」メソッドは、引数に指定したURLを、インターネットエクスプローラに設定するメソッドです。ここに、コンボボックスコントロールの［Text］プロパティの値を設定します。

コンボボックスコントロールには、［List］プロパティと［Text］プロパティにHTMLファイルのURLを設定していますから、ユーザーがクリックしたリストボックスの内容を、Navigator2メソッドの引数に渡してあげます。

```
Ie.Navigate2 Combo1.Text
```

Navigate2メソッドの書式

```
object.Navigate2 URL [Flags,] [TargetFrameName,]
    [PostData,] [Headers]
```

object	コントロールするオブジェクトを指定します。
URL	URLあるいはファイルのパスを指定します。
Flags	省略可能。履歴に追加するかどうか、キャッシュを読み書きするか、などを定数で指定します。定数はヘルプを参照してください。

TargetFrameName	省略可能。URLのウィンドウの表示位置を決定します。定数はヘルプを参照してください。
PostData	省略可能。HTTP POST処理中にサーバーへ送るデータを指定します。
Headers	省略可能。サーバーに送るHTTPヘッダーを指定します。

! ワンポイントアドバイス

「Navigate2」メソッドは、インターネットエクスプローラ4.0x以降に有効なメソッドです。バージョン3.0xを操作する場合は、「Navigate」メソッドを使用してください。「Navigate」メソッドは、Ver3.0x、4.0x、5.0xのいずれでも使用可能です。

5 ［Visible］プロパティは、インターネットエクスプローラを表示するかどうかを設定するプロパティです。値は論理値で、「True」か「False」を設定します。この［Visible］プロパティに「True」を設定した時点で、インターネットエクスプローラのインスタンスが、画面に表示されます。そして、Navigator2メソッドで設定されたURLのHTMLファイルが、ブラウザに表示されます。

```
Ie.Visible = True
```

また、［StatusText］プロパティに文字列を設定すると、インターネットエクスプローラ4.0では、ステータスバーにその文字を表示できます。

```
Ie.StatusText = "Hello!"
```

そして、［ブラウザの表示切り替え］ボタンと［ブラウザを閉じる］ボタンを有効にします。この2つのボタンは、フォームデザイン時に無効にしていました。インスタンスが作成されて、これらのボタンが有効になるようにします。

［Visible］プロパティに「True」を設定すると表示される

```
ViewBtn.Enabled = True
CloseBtn.Enabled = True
```

6　［ブラウザの表示切り替え］ボタンでは、コンボボックスの選択項目を、再度「Navigate2」メソッドの引数に当てて実行します。

```
Private Sub ViewBtn_Click()
    Ie.Navigate2 Combo1.Text
End Sub
```

また、［ブラウザを閉じる］ボタンでは、インターネットエクスプローラの［Visible］プロパティに「False」を設定し、非表示にします。表示を非表示にするだけですから、作成したインスタンスはそのまま保持されます。

```
Private Sub CloseBtn_Click()
    Ie.Visible = False
End Sub
```

なお、これらの処理は、複数のインスタンスを作成した場合は、最後に作成したインスタンスに対してのみ有効です。なぜなら、オブジェクト変数「Ie」には、最後に作成したインスタンスのオブジェクト名が格納されるからです。

7　以上で完成です。実際に動作させてください。インターネットエクスプローラそのものは、通常のブラウザと同じように、ホームページにアクセスでき、また作成したプログラムからもURLを変更して表示できます。

まとめ

インターネットエクスプローラをオブジェクトとして、自分のプログラムから自由に操作できます。このテクニックは、新しいブラウザを作成することなく、既存のオブジェクトを上手に活用するので、開発の手間が省けます。
「CreateObject」関数の使い方で注意するのは、戻り値がオブジェクトなので、必ずオブジェクト変数にSetステートメントを使って格納することです。InternetExplorerオブジェクトの保有するプロパティやメソッドの一覧は、P.303のコラムに記載していますので、そちらを参照してください。

練習問題

Q1　作成したプログラムを、インスタンスを1つだけしか生成できないように修正してください。

解答は巻末に→

COLUMN ・・・・・

Internet Client SDK

このレッスンで紹介している内容は、Visual Basicのオンラインヘルプには記載されていません。これらの情報は、すべて「MSDN」から出されているドキュメント（すべて英文）に記載されています。これらは、次の方法で入手することができます。

①Microsoftが運営している、開発者向け技術情報サービス「MSDN」（Microsoft Developer Network）から発行されている「MSDN Library」、またはVisual Studio6.0に付属する「MSDN Library」、またはOffice2000 Developer Editionに付属する「MSDN Library」に記載されています。

②最新情報はインターネット上に公開されています。次のURLにWebBrowserコントロールについての詳細なドキュメントがありますので参照してください。

http://msdn.microsoft.com/workshop/browser/

※これらのURLは、1999年11月現在確認したURLです。この日付以降はURLが変更されている場合もあるのでご了承ください。

MSDNのホームページ

「msdn Library」に記述されているWebBrowser Objectのリファレンス

COLUMN

インターネットエクスプローラオブジェクトが持っているプロパティとメソッド、イベントの一覧

<プロパティ>

AddressBar、Application、Busy、Container、Document、FullName、FullScreen、Height、HWND、Left、LocationName、LocationURL、MenuBar、Name、Offline、Parent、Path、ReadyState、RegisterAsBrowser、RegisterAsDropTarget、Resizable、Silent、StatusBar、StatusText、TheaterMode、ToolBar、Top、TopLevelContainer、Type、Visible、Width

<メソッド>

ClientToWindow、ExecWB、GetProperty、GoBack、GoForward、GoHome、GoSearch、Navigate、Navigate2、PutProperty、QueryStatusWB、Quit、Refresh、Refresh2、ShowBrowserBar、Stop

<イベント>

BeforeNavigate2、CommandStateChange、DocumentComplete、DownloadBegin、DownloadComplete、NavigateComplete2、NewWindow2、OnFullScreen、OnMenuBar、OnQuit、OnStatusBar、OnTheaterMode、OnToolBar、OnVisible、ProgressChange、PropertyChange、StatusTextChange、TitleChange

Lesson 23

第7日 2時限目

カスタムWebブラウザの作成

Lesson23.vbp
VB6CD2
Lesson23

Internet Explorer Control Padの作成

今度は、インターネットエクスプローラのツールバーやメニューなどを非表示にし、HTMLの表示のみを行う設定をします。一方、作成したプログラムに［戻る］、［進む］などのボタンを組み込み、完全にプログラムからインターネットエクスプローラの制御を行うようにします。

今回作成するプログラム

WWWブラウザのメニューとツールバーを非表示にし...

htm - Microsoft Internet Explorer

このブラウザは、Visual Basicから起動しました。

Internet Explorer Control Pad

c:¥ie.htm

新しいブラウザの起動

Control

Back　Stop　Fwd

Reload　Home

ブラウザの表示切り替え

ブラウザを閉じる

プログラムから操作を行う

マイ コンピュータ

POINT

インターネットエクスプローラの外観を制御するプロパティを設定します。また、インターネットエクスプローラにあるツールバーと同じ機能を持ったボタンをプログラムに組み込み、インターネットエクスプローラのすべての操作をプログラムで行うようにします。

コントロールとプロパティの設定

※Lesson22のQ&Aで作成した、「Q&A-22.vbp」を開いてください。

1 ツールパレットの［Frame］を使って、フレームコントロールを配置する

Internet Explorer Control Pad
c:¥ie.htm
新しいブラウザの起動
ブラウザの表示切り替え
ブラウザを閉じる
Frame1

1 斜めにドラッグして配置

2 ［Caption］プロパティを「Control」に変更する

プロパティ - Frame1
Frame1 Frame
全体 ｜ 項目別

(オブジェクト名)	Frame1
Appearance	1 - 3D
BackColor	&H8000000F&
BorderStyle	1 - 実線
Caption	Control
ClipControls	True
DragIcon	(なし)
DragMode	0 - 手動

1 ここを「Control」に変更

3 フレームコントロールの中にコマンドボタンを5つ配置し、プロパティを設定する

ボタン	オブジェクト名	Caption	Enabled
Command1	BkBtn	Back	False
Command2	StopBtn	Stop	False
Command3	FwdBtn	Fwd	False
Command4	RelBtn	Reload	False
Command5	HomeBtn	Home	False

Q&A-22.vbp

「Q&A-22.vbp」ファイルは、付録CD-ROMの「Q&A」フォルダにも収録されています。

また、Lesson22で使った同じHTMLファイルも使用します。

コマンドボタンを配置する

コマンドボタンを配置するには、ツールパレットの［CommandButton］を使います。

完成したフォーム

コードの記述

Form1:RunBtn-Click

Lesson23 - Form1 (ｺｰﾄﾞ)

RunBtn / Click

```
Private Sub RunBtn_Click()
    Set Ie = CreateObject("InternetExplorer.application")

    'NavigateメソッドはIE Ver4.0、3.0ともに有効
    Ie.Navigate Combo1.Text

    'このレッスンで追加したコード
    With Ie
        .ToolBar = False
        .MenuBar = False
        .Height = 400
        .Width = 400
        .Top = 100
        .Left = 300
    End With

    Ie.Visible = True
    Ie.StatusText = "Hello!"

    ViewBtn.Enabled = True
    CloseBtn.Enabled = True
    RunBtn.Enabled = False
    BkBtn.Enabled = True
    StopBtn.Enabled = True
    FwdBtn.Enabled = True
    RelBtn.Enabled = True
```

```
    HomeBtn.Enabled = True
End Sub
```

Form1:ViewBtn-Click

Lesson23 - Form1 (ｺｰﾄﾞ)

ViewBtn | Click

```
Private Sub ViewBtn_Click()
    On Error Resume Next
    Ie.Navigate Combo1.Text
End Sub
```

Form1:CloseBtn-Click

Lesson23 - Form1 (ｺｰﾄﾞ)

CloseBtn | Click

```
Private Sub CloseBtn_Click()
    On Error Resume Next
    Ie.Quit
    ViewBtn.Enabled = False
    CloseBtn.Enabled = False
    RunBtn.Enabled = True

    BkBtn.Enabled = False
    StopBtn.Enabled = False
    FwdBtn.Enabled = False
    RelBtn.Enabled = False
    HomeBtn.Enabled = False
End Sub
```

Form1:BkBtn-Click

Lesson23 - Form1 (ｺｰﾄﾞ)

BkBtn | Click

```
Private Sub BkBtn_Click()
    On Error Resume Next
    Ie.GoBack
End Sub
```

Form1:StopBtn-Click

Lesson23 - Form1 (ｺｰﾄﾞ)

StopBtn | Click

```
Private Sub StopBtn_Click()
    On Error Resume Next
    Ie.Stop
End Sub
```

Form1:FwdBtn-Click

Lesson23 - Form1 (コード)

FwdBtn | Click

```
Private Sub FwdBtn_Click()
    On Error Resume Next
    Ie.GoForward
End Sub
```

Form1:RelBtn-Click

Lesson23 - Form1 (コード)

RelBtn | Click

```
Private Sub RelBtn_Click()
    On Error Resume Next
    Ie.Refresh
End Sub
```

Form1:HomeBtn-Click

Lesson23 - Form1 (コード)

HomeBtn | Click

```
Private Sub HomeBtn_Click()
    On Error Resume Next
    Ie.GoHome
End Sub
```

解説

1　インターネットエクスプローラオブジェクトには、ツールバーやメニューの表示と非表示を設定するプロパティや、ウィンドウのサイズと位置を設定するプロパティを持っています。また、ツールバーにある「戻る」「進む」「中止」「更新」「ホーム」などのボタンと同じ動作をするメソッドも持っています。このプログラムでは、インターネットエクスプローラのメニューとツールバーを非表示にして、プログラムから操作を行うようにします。

メニューとツールバーを非表示にし…

このブラウザは、Visual Basicから起動しました。

プログラムから操作を行う

2　最初に、インターネットエクスプローラのインスタンスを表示する際の、表示の形態と位置を設定します。これらは、インスタンスを作成してから、［Visible］プロパティで表示する前に設定します。「ToolBar」「MenuBar」は、それぞれインターネットエクスプローラのツールバーとメニューを表示するかどうかを設定します。「True」で表示、「False」で非表示になります。また、「Height」「Width」はそれぞれブラウザの高さと幅、「Top」「Left」はブラウザを表示する画面の位置を設定します。単位はピクセルになります。

```
Private Sub RunBtn_Click()
    Set Ie = CreateObject("InternetExplorer.application")
    Ie.Navigate Combo1.Text

    With Ie
        .ToolBar = False
        .MenuBar = False
        .Height = 400
        .Width = 400
        .Top = 100
        .Left = 300
    End With

    Ie.Visible = True
```

3　次に、フォームに配置した、インターネットエクスプローラを操作する5つのボタンの［Enabled］プロパティを、「True」に設定して有効にします。プログラム起動時は、これらのボタンは無効になっていますが、インスタンスを作成した時点で操作可能にします。

```
BkBtn.Enabled = True
StopBtn.Enabled = True
FwdBtn.Enabled = True
RelBtn.Enabled = True
HomeBtn.Enabled = True
```

そして、［ブラウザの表示切り替え］ボタンにエラー処理ステートメント「On Error Resume Next」を組み込んでおきます。これは、フォームの「ブラウザを閉じる」ボタンを使わずに、インターネットエクスプローラの［閉じる］ボタンでブラウザを閉じて、［ブラウザの表示切り替え］ボタンをクリックすると「オートメーションエラー」が発生してしまいます。オブジェクト変数にはオブジェクトが格納されているのに、実際にはインスタンスが存在していないために起こるエラーです。

このエラーを回避するために、「On Error Resume Next」を使用します。これで、エラーが起きても、インターネットエクスプローラに対するページ表示の処理はせずに、プロシージャを終了します。

```
Private Sub ViewBtn_Click()
    On Error Resume Next
    Ie.Navigate Combo1.Text
End Sub
```

4　［ブラウザを閉じる］ボタンのClickイベントプロシージャにも、コードを追加します。まず、［ブラウザの表示切り替え］ボタンの処理と同様、「On Error Resume Next」ステートメントを組み込みます。やはり、インターネットエクスプローラの［閉じる］ボタンでブラウザを閉じてしまったのに、［ブラウザを閉じる］ボタンをクリックしてしまうと、同じようにエラーを起こしてしまいますから、これを回避します。

```
Private Sub CloseBtn_Click()
    On Error Resume Next
    Ie.Quit

    ViewBtn.Enabled = False
    CloseBtn.Enabled = False
    RunBtn.Enabled = True
```

そして、フォームにあるインターネットエクスプローラを操作する5つのボタンを、無効にします。

```
BkBtn.Enabled = False
StopBtn.Enabled = False
FwdBtn.Enabled = False
RelBtn.Enabled = False
HomeBtn.Enabled = False
```

5　最後に、インターネットエクスプローラを操作する5つのボタンの処理を作成します。それぞれ、専用のメソッドを実行するだけです。いずれも、「On Error Resume Next」ステートメントを組み込み、エラーに対処します。たとえば、［Back］ボタンは、戻るページがないのにこのメソッドを実行するとエラーになってしまいます。そこで、エラーステートメントでこれを回避するようにします。

```
Private Sub BkBtn_Click()
    On Error Resume Next
    Ie.GoBack
End Sub
```

それぞれのメソッドは、インターネットエクスプローラのツールバーにあるボタンと同じ動作をします。各メソッドの機能は、次のとおりです。

メソッド名	機能
GoBack	現在のページより１つ前に表示したページを表示します。
GoForward	現在のページより次のページを表示した場合は、そのページを再表示します。
Stop	表示中の処理を停止します。
Refresh	現在のページを再度読み込みます。
GoHome	ホームに設定しているページを表示します。

まとめ

インターネットエクスプローラのプロパティやメソッドは、まだほかにもいくつか用意されていますが、とりあえずWebをナビゲートするには、これらが使えれば十分でしょう。エラー処理は、ここでは一番簡単な方法を組み込んでいます。プログラムの動作にあわせて自分なりに改良してください。Navigateメソッドは基本的にNavigate2メソッドと書式は同じですがインターネットエクスプローラのバージョン3.0x以上から使用できます。Navigate2は4.0x以上のみサポートです。

これらのプロパティやメソッドは、インターネットエクスプローラの将来のバージョンで変更される可能性があります。開発に関する技術情報は、つねに最新のものを把握するように心がけておきましょう。

練習問題

Q1　作成したプログラムのフォームに、インターネットエクスプローラで表示している現在のURLを、ラベルコントロールで表示するように修正してください。

解答は巻末に→

ワンポイントアドバイス

プログラム終了時に、自動的にインスタンスのインターネットエクスプローラを閉じたい場合は、フォームの「Unload」イベントプロシージャに、インターネットエクスプローラオブジェクトの「Quit」メソッドを実行する処理を作成しておきます。

```
Private Sub Form_Unload(Cancel As Integer)
    On Error Resume Next
    Ie.Quit
End Sub
```

Lesson 24

第７日
３時限目

カスタム
Webブラウザ
の作成

Lesson24.vbp
VB6CD2
Lesson24

カスタムWebブラウザの作成

これまでのレッスンでは、インターネットエクスプローラのインスタンスを作成して操作するテクニックを使用しました。今度は、フォームにWebブラウザのコントロールを組み込んで使用するテクニックを紹介します。コンピュータにインターネットエクスプローラを組み込むと、いっしょに「WebBrowser」コントロールが組み込まれます（インターネットエクスプローラも、このコントロールを使用して組み立てられています）。このコントロールをフォームに組み込んで使用することができます。

今回作成するプログラム

My Browser Ver1.0
戻る 停止 進む 更新 ホーム
表示完了
http://www.big.or.jp/~seto/
Go
ページ表示
SETO HOME
ようこそ、瀬戸 遥のホームページへ!
最新 情報

HTMLを表示する、独自のWebブラウザを作成

POINT

フォームにWebBrowserコントロールを組み込み、HTMLを表示する独自のWebブラウザを作成します。

コントロールとプロパティの設定

※新しいプロジェクトファイル（標準EXE）を用意します。

1 ツールパレットに次のコンポーネントを追加する

- Microsoft Internet Controls
- Microsoft Windows Common Controls6.0

IE4.0/5.0の場合に表示されるアイコン

2 フォームの［Caption］プロパティを「My Browser Ver1.0」に変更する

プロパティ－Form1

Form1 Form

全体 | 項目別

Appearance	1 – 3D
AutoRedraw	False
BackColor	&H8000000F&
BorderStyle	2 – 可変
Caption	My Browser Ver1.0
ClipControls	True
ControlBox	True
DrawMode	13 – Copy Pen
DrawStyle	0 – 実線（中心線）
DrawWidth	1

1 ここを変更

コンポーネントを追加する

コンポーネントを追加する方法については、「Lesson5 ツールバーコントロールの組み込み」を参照してください。

3 ツールパレットの［ComboBox］を使って、コンボボックスコントロールを配置する

4 ［List］プロパティを次のように設定する

5 ［Text］プロパティを「c:¥ie.htm」に設定する

6 コマンドボタンを配置し、[(オブジェクト名)] プロパティと [Caption] プロパティを設定する

プロパティ	設定
(オブジェクト名)	GoBtn
Caption	Go

7 ツールバーコントロールを配置し、[BorderStyle] プロパティを「1 －実線」に設定する

8 [プロパティページ] の [ボタン] タブで、[ボタンの挿入] を使ってボタンを５つ配置する

インデックス	キャプション	キー
1	戻る	back
2	停止	stop
3	進む	fwd
4	更新	reload
5	ホーム	home

コマンドボタンを配置する

コマンドボタンを配置するには、ツールパレットの [CommandButton] を使います。

ツールバーを配置する

ツールバーを配置するには、ツールパレットの [Toolbar] を使います。

[プロパティページ] ダイアログ

[プロパティページ] ダイアログボックスを表示するには、コントロールを選択し、プロパティウィンドウの [(プロパティページ)] をクリックし、右欄の [...] ボタンをクリックします。

[Align] プロパティ

「0 − vbAlignNone」を設定すると、ツールバーコントロールのサイズ変更ハンドルをドラッグして、サイズを調整できるようになります。

9 [Align] プロパティを「0 − vbAlignNone」に設定し、サイズを小さくする

フレームコントロールを配置する

フレームコントロールを配置するには、ツールパレットの［Frame］ を使います。

10 フレームコントロールを配置し、[Caption] プロパティを「ページ表示」に変更する

11 ツールパレットの［WebBrowser］を使って、WebBrowser コントロールを配置する

My Browser Ver1.0
戻る 停止 進む 更新 ホーム
c:¥ie.htm Go
ページ表示

1 斜めにドラッグして配置

12 [DragMode] プロパティを「0 – vbManual」に設定する

プロパティ - WebBrowser1
WebBrowser1 WebBrowser
全体 項目別

(オブジェクト名)	WebBrowser1
AddressBar	True
CausesValidation	True
DragIcon	(なし)
DragMode	0 – vbManual
FullScreen	0 – vbManual
Height	1 – vbAutomatic
HelpContextID	0
index	
Left	240

1 ここをクリック
2 ここをクリック
3 ここをクリック

13 ツールパレットの［Label］A を使って、ラベルコントロールを配置する

My Browser Ver1.0
戻る 停止 進む 更新 ホーム
c:¥ie.htm Go
ページ表示

1 斜めにドラッグして配置

ドラッグモードプロパティ

ドラッグモード（DragMode）プロパティが「1 – vbAutomatic」になっているとコントロールをマウスで操作することができなくなり、スクロールバーなどが使えなくなります。

14 [BorderStyle] プロパティを「1-実線」に設定し、[Caption] プロパティを削除する

プロパティを設定したラベルコントロール

コードの記述

Form1:Form-Resize

Lesson24 - Form1 (コード)

Form / Resize

```
Private Sub Form_Resize()
    If WindowState <> vbMinimized Then
        With Frame1
            .Height = Form1.Height - 1800
            .Width = Form1.Width - 500
        End With

     With WebBrowser1
            .Height = Frame1.Height - 500
            .Width = Frame1.Width - 500
        End With
    End If
End Sub
```

Form1:GoBtn-Click

Lesson24 - Form1 (コード)

GoBtn / Click

```
Private Sub GoBtn_Click()
    On Error Resume Next
    WebBrowser1.Navigate Combo1.Text
End Sub
```

Form1:WebBrowser1-DownloadBegin

Lesson24 - Form1 (コード)

WebBrowser1 | DownloadBegin

```
Private Sub WebBrowser1_DownloadBegin()
    Label1.Caption = "転送開始"
End Sub
```

Form1:WebBrowser1-DownloadComplete

Lesson24 - Form1 (コード)

WebBrowser1 | DownloadComplete

```
Private Sub WebBrowser1_DownloadComplete()
    Label1.Caption = "表示完了"
End Sub
```

Form1:WebBrowser1-ProgressChange

Lesson24 - Form1 (コード)

WebBrowser1 | ProgressChange

```
Private Sub WebBrowser1_ProgressChange(ByVal Progress As Long, _
    ByVal ProgressMax As Long)
    Dim i As Double

    If Progress > 0 Then
        i = (Progress / ProgressMax) * 100
        If i >= 100 Then
            i = 100
        End If
        Label1.Caption = "現在転送中：" & i & " %"
    End If
End Sub
```

Form1:Toolbar1-ButtonClick

Lesson24 - Form1 (コード)

Toolbar1 | ButtonClick

```
Private Sub Toolbar1_ButtonClick(ByVal Button As _
    MSComctlLib.Button)
    On Error Resume Next
    Select Case Button.Key

        Case "back"
            WebBrowser1.GoBack
        Case "stop"
            WebBrowser1.Stop
```

```
        Case "fwd"
            WebBrowser1.GoForward
        Case "reload"
            WebBrowser1.Refresh
        Case "home"
            WebBrowser1.GoHome
    End Select
End Sub
```

解 説

1 WebBrowserコントロールは、インターネットエクスプローラとまったく同様の機能を提供するコントロールです。コントロールが保有するプロパティやメソッドは、インターネットエクスプローラ・オブジェクトとほぼ同様のものです。また、コントロールとして操作できるために、いくつかのイベントを使ったプログラミングが可能になっています。

WebBrowserコントロールをプログラムに組み込むには、Visual Basicのツールパレットに、「Microsoft Internet Controls」を追加します。ただし、このコントロールをフォームに配置しても、配置した時点ではなにも表示するものがありませんので、コントロールは見えづらくなります。また、コントロールなので、ごく一部のプロパティについては、プロパティウィンドウで初期設定が行えます。

プロパティ - WebBrowser1

WebBrowser1 WebBrowser

全体 | 項目別

(オブジェクト名)	WebBrowser1
AddressBar	True
CausesValidation	True
DragIcon	(なし)
DragMode	0 - vbManual
FullScreen	False
Height	3255
HelpContextID	0
index	
Left	240
MenuBar	True

WebBrowserコントロールのプロパティウィンドウ

WebBrowserコントロール自身は、コンピュータにインターネットエクスプローラをインストールした時点でいっしょに組み込まれます。コントロールをインストールするための特別な操作は必要ありません。

2 このプログラムは、インターネットエクスプローラと同様の基本機能を持ったWebブラウザとして作成します。簡単にプログラムの動作を説明します。

①コンボボックスコントロールのリストボックスからURLを選択するか、テキスト入力エリアにURLを入力し、[Go] ボタンをクリックすると、WebBrowserコントロールの「Navigate」メソッドを使って、HTMLを表示します。

②HTMLは、フォームに配置したWebBrowserコントロールに表示されます。コントロール内にHTMLドキュメントが表示しきれない場合は、WebBrowserコントロールには自動的にスクロールバーが表示されます。

③指定したHTMLファイルとそのイメージファイルが転送され始めると、ラベルコントロールに転送状況が表示されます。

④転送が完了すると、ラベルコントロールの表示が「表示完了」に変化します。これらの処理は、WebBrowserコントロールのイベントを使って行います。

⑤ツールバーコントロールに組み込んだボタンは、それぞれページを表示したり停止させるなどの処理を行います。

3 では、コードの作成を始めましょう。まず、WebBrowserコントロールは、インターネットエクスプローラのインスタンスとして表示するわけではありませんから、HTMLの表示サイズはコントロールのサイズによって決まってしまいます。そこで、フォームのサイズを変えると自動的にWebBrowserコントロールのサイズも変わるような処理を組み込みます。この処理は、フォームの「Resize」イベントに作成します。

```
Private Sub Form_Resize()
    If WindowState <> vbMinimized Then
        With Frame1
            .Height = Form1.Height - 1800
            .Width = Form1.Width - 500
        End With

        With WebBrowser1
            .Height = Frame1.Height - 500
            .Width = Frame1.Width - 500
        End With
    End If
End Sub
```

4 WebBrowserコントロールにHTMLを表示する処理は、[Go] ボタンで行います。[Go] ボタンのClickイベントプロシージャでは、コンボボックスコントロールの [Text] プロパティからURLを受け取って、「Navigate」メソッドを実行します。このとき、存在しないURLが入力されると、「Navigate」メソッドでエラーが発生しますから、これを「On Error Resume Next」ステートメントでトラップします。

```
Private Sub GoBtn_Click()
    On Error Resume Next
    WebBrowser1.Navigate Combo1.Text
End Sub
```

HTMLの表示は、インターネットエクスプローラのインスタンスを使うわけではありませんから、[Visible] プロパティの操作をする必要はなく、「Navigate」メソッドの実行だけでHTMLが表示されます。

5 指定したURLからHTMLファイルやイメージファイルの転送が開始されると、WebBrowserコントロールには「DownloadBegin」イベントが発生します。そこで、このイベントプロシージャで、ラベルコントロールに「転送開始」の文字列を設定します。

```
Private Sub WebBrowser1_DownloadBegin()
    Label1.Caption = "転送開始"
End Sub
```

「転送開始」の文字列を設定する

また、ファイルの転送が終了すると、WebBrowserコントロールには「DownloadComplete」イベントが発生します。そこで、ラベルコントロールに「表示完了」の文字を設定します。

```
Private Sub WebBrowser1_DownloadComplete()
    Label1.Caption = "表示完了"
End Sub
```

「表示完了」の文字列を設定する

6　サーバからファイルが転送され始めると、WebBrowserコントロールには、「ProgressChange」イベントが発生します。このイベントプロシージャを使うと、転送中の状況をフォームに表示できます。

「ProgressChange」イベントプロシージャは、2つの引数を持ちます。1つは「Progress」で、転送されてきたデータのバイト数が格納されます。もう1つは、「ProgressMax」で、「Progress」に対する最大値を設定します。この引数は、自分で自由に設定できます（デフォルトで「1,000,000」バイト（1MB）になっています）。「ProgressChange」イベントは、データがサーバから転送させるたびに発生し、トータルの転送バイト数が引数「Progress」に格納されます。そこで、このプログラムでは、以下の式を作って、転送バイト数を「%」で表示するようにします。「ProgressMax」の値は、「1,000,000」バイト（1MB）のまま使用します。

```
i = (Progress / ProgressMax) * 100
```

すでに転送されているデータを再度転送した場合は、引数「Progress」には「-1」が格納されます。この場合は、式がエラーになります。また、転送データの合計バイト数が「ProgressMax」を超えることがありますので、これらの処理をIfステートメントで設定しておきます。

```
Private Sub WebBrowser1_ProgressChange _
    (ByVal Progress As Long, ByVal ProgressMax As Long)
    Dim i As Double

    If Progress > 0 Then
        i = (Progress / ProgressMax) * 100
        If i >= 100 Then
            i = 100
        End If
        Label1.Caption = "現在転送中：" & i & " %"
    End If
End Sub
```


転送バイト数が%で表示される

ワンポイントアドバイス

このイベントプロシージャの処理は、ローカルコンピュータのファイルに対しては、データ転送が早すぎて正常に動作しない場合があります。

7 最後に、ツールバーコントロールのボタンの処理を行います。これは、インターネットエクスプローラオブジェクトと同じメソッドを使用します。それぞれのボタンに設定した「キー」を使って、Select Caseステートメントで処理を作成します。

「GoHome」メソッドは、Windowsのレジストリを使って、インターネットエクスプローラで設定されている、ホームのHTMLの場所を探し出して表示します。「On Error Resume Next」ステートメントで、エラー処理を組んでおくのを忘れないでください。

```
Private Sub Toolbar1_ButtonClick(ByVal Button As _
    MSComctlLib.Button)
    On Error Resume Next
    Select Case Button.Key

        Case "back"
            WebBrowser1.GoBack
        Case "stop"
            WebBrowser1.Stop
        Case "fwd"
            WebBrowser1.GoForward
        Case "reload"
            WebBrowser1.Refresh
        Case "home"
            WebBrowser1.GoHome
    End Select
End Sub
```

まとめ

　WebBrowserコントロールは、インターネットエクスプローラの機能を自分のプログラムに組み込むことができます。既存のリソースを使ったプログラミングができるため、実に簡単にブラウザ機能を実装できます。

　また、WebBrowserコントロールは、ActiveXコントロールなので、Microsoft ExcelやWordのドキュメントファイルを表示することもできます。ブラウザに固定せず、自由な発想で楽しいプログラムを開発してください。

インターネットエクスプローラ4.0のWebBrowserコントロールでExcelのワークシートを表示

ワンポイントアドバイス

WebBrowserコントロールのメソッド、プロパティ、イベントの使い方についても、「Internet Client SDK」に詳しいドキュメントが用意されています（P.302参照）。

WebBrowserコントロールの「ExecWB」メソッドを使用すると、コントロールで表示するファイルを開いたり、表示しているHTMLの保存や印刷ができるようになります。

ExecWBメソッドの書式

```
object.ExecWB nCmdID, nCmdExecOpt, [pvaIn], [pvaOut]
```

object	対象となるオブジェクトを指定します。
nCmdID	実行するコマンド名かID番号（長整数）を指定します。コマンド名についてはヘルプを参照してください。
nCmdExecOpt	コマンドのオプション名かID番号（長整数）を指定します。オプション名についてはヘルプを参照してください。
pvaIn	省略可。コマンドの入力変数を指定します。
pvaOut	省略可。コマンドの出力変数を指定します。

練習問題

Q1 現在表示しているURLを、フォームのキャプションに表示してください。

Q2 コンボボックスコントロールにURLを入力して、Enterキーを押した時点で、WebBrowserコントロールにHTMLを表示するように修正してください。

Q3 フォームに［SaveAs］（ファイルに保存）ボタンと［保存］ツールバーを設け、表示中のHTMLファイルを保存できるようにしてください。

解答は巻末に→

Let's VBSchool!!

第8日 (Lesson25 ～ Lesson27)
Win32APIを使う

Visual Basicのコントロールは、Windowsのすべての機能を網羅してはいません。そこで、Visual Basicが持っていないオペレーティングシステム（OS）の機能を、プログラムに組み込む方法を紹介します。Windowsには、Windowsの機能を自由に操作するための特別なインターフェイス「Application Programming Interface（API）」があります。Windowsの上で動作するアプリケーションプログラムは、すべてこのAPIを呼び出して組み立てられています。今日のレッスンでは、APIに関する基礎的な知識と、Visual Basicのプログラムで APIを呼び出す方法について、理解を深めていきます。

今日のレッスンで解説する項目

- ❑ Win32APIに関する初歩的な知識
- ❑ API関数の使い方の基本
- ❑ API関数「GetComputerName」の使い方
- ❑ 関数が返してきた値をコントロールに反映させるテクニック
- ❑ APIに関する情報の入手方法
- ❑ APIのVisual Basicへの実装方法
- ❑「Declare」ステートメントの使い方
- ❑ 標準モジュールの使い方
- ❑ APIの引数とVisual Basicのデータ型
- ❑ APIビューアの使い方
- ❑「SetComputerName」関数を使う
- ❑ APIの戻り値とエラー処理
- ❑ 固定長の文字列型変数の宣言
- ❑ API関数「GlobalMemoryStatus」を使う
- ❑ ユーザー定義のデータ型の定義と宣言、値の取得

Lesson 25

第８日
１時限目

Win32APIを使う

Lesson25.vbp
VB6CD2
Lesson25

APIの機能と特徴

このプログラムは、「GetComputerName」というAPI関数を呼び出し、ラベルコントロールに現在動作しているコンピュータの名前を表示するプログラムです。このプログラム作成を通じて、APIの機能と特徴について理解をしていきます。

今回作成するプログラム

このボタンをクリックすると...

Form1

使用しているコンピュータ名

PCG-880 SERIES

ここに使用しているコンピュータの名前が表示される

POINT

API関数「GetComputerName」の使い方と、返してきた値をコントロールに反映させるテクニックがポイントです。APIそのものは、とても簡単なAPIで、使用しているコンピュータの名前を返すAPIです。ただし、このようなメソッドや関数は、Visual Basicにはありません。Visual Basicの持っていない機能をプログラムに組み込みたい場合にAPIを使用します。

コントロールとプロパティの設定

※新しいプロジェクトファイル（標準EXE）を用意します。

1 コマンドボタンを1つ配置し、[Caption] プロパティを「使用しているコンピュータ名」に変更する

Form1

使用しているコンピュータ名

配置して［Caption］を変更したコマンドボタンコントロール

2 ツールパレットの［Label］ A を使ってラベルコントロールを配置し、次のプロパティを設定する

プロパティ	設定
BorderStyle	1 －実線
Caption	（なし）
Alignment	2 －中央揃え
Font	［サイズ］を「10」

Form1

使用しているコンピュータ名

配置してプロパティを設定したラベルコントロール

コマンドボタンを配置する

コマンドボタンを配置するには、ツールパレットの［CommandButton］を使います。

フォントのサイズを変更する

フォントのサイズを変更するには、［Font］プロパティの［...］ボタンをクリックし、［フォントの指定］ダイアログボックスの［サイズ］ボックスで「10」を選択します。

標準モジュールを挿入する

［プロジェクト］メニューの［標準モジュールの追加］を選択して、標準モジュールを挿入することもできます。

3 プロジェクトに標準モジュールを挿入する

Lesson25 - Microsoft Visual Basic [デザイン]

ファイル(F) 編集(E) 表示(V) プロジェクト(P) 書式(O) デバッグ(D) 実行

フォーム モジュール(F)
MDI フォーム モジュール(I)
標準モジュール(M)
クラス モジュール(C)
ユーザー コントロール(U)
プロパティ ページ(R)

1 ここをクリック
2 ここをクリック

標準モジュールの追加

新規作成 | 既存のファイル

標準モジュール

開く(O)
キャンセル
ヘルプ(H)

次回からこのダイアログ ボックスを表示しない(D)

3 ここをクリック
4 ここをクリック

コードの記述

Module1:(General)-(Declarations)

Lesson25 - Module1 (コード)

(General) | (Declarations)

```
Declare Function GetComputerName Lib "kernel32" _
    Alias "GetComputerNameA" (ByVal lpBuffer As String, _
    nSize As Long) As Long
```

Form1:Command1-Click

Lesson25 - Form1 (コード)

Command1 | Click

```
Private Sub Command1_Click()
    Dim name As String * 16

    GetComputerName name, 16
    Label1.Caption = name
End Sub
```

解　説

1　まずはじめに、APIについて簡単に説明しておきます。

私たちは、Windowsの持っている大変膨大な機能を、1つ1つ呼び出して使うことができます。これらのWindowsの機能は、いくつものライブラリファイルに格納されています。ライブラリファイルは拡張子が「DLL」のファイルで、Windows(95またはNT）のディレクトリ（フォルダ）に保存されています。APIとは、これらのライブラリファイルから必要な機能を、外部（他のアプリケーションプログラム）から使えるようにした、関数のことです。代表的なライブラリファイルと利用できる機能は、次のものがあります。

ライブラリファイル	機能
Kernel32.dll （システムサービス・インターフェイス）	ファイルシステムやメモリ管理、システム情報などを管理するライブラリで、これらの機能を提供するAPIが用意されています。
User32.dll （ユーザーサービス・インターフェイス）	ウィンドウやダイアログボックス、コントロールなどのユーザーインターフェイスを担当するライブラリです。
Gdi32.dll （グラフィックデバイス・インターフェイス）	ペンやブラシの設定、ビットマップや図形の作成など、画面への描画を担当するライブラリです。

他にも、いくつかのライブラリファイルが持っているAPIを使用することができます。Win32APIでは、合計1000個以上のAPIを、プログラマが自由に使用できるようになっています。

ワンポイントアドバイス

Win32APIでは、32ビットのWindowsに対応したAPIのことで、Windows3.1のような16ビットのWindowsに対応していません。本稿では、Windows95/98/WindowsNTを操作対象にしていますので、Win32APIを使用します。

2 「C／C++言語」を使ってWindowsのプログラミングを行ったことのない人は、いったいどのような種類のAPIがあって、どんなときにどのAPIを使えばいいのか、さっぱりわからないと思います。

APIに関するドキュメントは、Windowsのアプリケーションやドライバを開発するキット、「Win32 Software Development Kit」（通称 Win32 SDK）に同梱されています。この中にある「Win32 SDK Help」にAPIのリファレンスが掲載されています（ただしすべて英文です）。この「Win32 SDK」は、一般には市販されておらず、Microsoft社が提供している技術情報サービス「Microsoft Developer Network」のメンバー向けに配布されている「ライブラリCD-ROM」の中に収録されています。

Win32 SDK
オンラインヘルプ

「MSDN」の詳細については、「http://www.microsoft.com/japan/developer/」もしくは「http://www.microsoft.com/msdn/」を参照のこと。

また、『Windows95 APIバイブル 1』（小社刊）に、使用できるWin32APIが詳しく解説されています。こちらは日本語のリファレンスで、サンプルプログラムも付属しています（ただしC言語用）ので、こちらも利用すると便利です。

「Windows95　APIバイブル1」（翔泳社）
本体価格9,600円
Windows95でサポートされている32ビットAPIの関数および関連メッセージなどを機能別に分類して解説しているプログラマ必携の書。

3 APIは、基本的には「C言語」で使うように作られています。たとえば、今回作成するプログラムで使用するAPI「GetComputerName」は、完全に「C言語」の関数として作成されています。

```
BOOL GetComputerName
   (
   LPTSTR  lpBuffer, // address of name buffer
   LPDWORD  nSize  // address of size of name buffer
  );
```

「GetComputerName」関数を実行すると、引数「lpBuffer」には、コンピュータ名を格納したメモリの、先頭のアドレスが格納されます。このアドレスを参照することで、メモリに格納されているコンピュータ名を取り出すことができます。

しかし、Visual Basicでは、「C言語」の関数をそのまま使うことができません。関数の構文が違いますし、引数のデータ型も違います。そこで、このAPIをVisual Basicでも使えるようにしたのが、「Declare」ステートメントによる外部プロシージャへの参照宣言です。「Declare」ステートメントを使うと、Visual Basicが使用するAPIが格納されているダイナミックリンク・ライブラリ（DLLファイル）を参照して、そのAPIをVisual Basicの「Function」形式に変換して、他のプロシージャで使用できるようにしてくれます。

この「GetComputerName」関数を、「Declare」ステートメントで宣言すると次のような構文になります。

```
Declare Function GetComputerName Lib "kernel32" _
                  ①                    ②
   Alias "GetComputerNameA" (ByVal lpBuffer As String, _
               ③                         ④1
   nSize As Long) As Long
        ④2
```

この記述をよく見ると、次の書式で書かれています（Declareステートメントの書式はP.350を参照）。

①関数名「GetComputerName」はそのままで、頭に「Function」キーワードがついています。これで、Visual Basicの関数形式のプロシージャとして使用できるようになります。

②関数名のあとには、APIが格納されているダイナミックリンクライブラリの名前を記述します。このAPIは、「kernel32.dll」の機能を使うためのインターフェイスを取りますので、「"kernel32"」と記述します。

③「Alias」は、この関数がライブラリの中で別の名前を持っている場合に使用します。このAPIは、ライブラリの中では「GetComputerNameA」という名前で使用されています。

④2つの引数は、それぞれ文字列型（String）と長整数型（Long）に変換されて宣言し直されています。このデータ型であれば、Visual Basicで使用できます。

このように、APIをVisual Basicで使う場合は、「Declare」ステートメントを用いて、Visual Basicで使える形に宣言し直してから使用します。

4 では、実際にコードを作成してみましょう。まず、プロジェクトに挿入した標準モジュール「Module1」では、Declareステートメントで API「GetComputerName」を宣言するだけです。

以下の宣言を、大文字小文字を正確に入力（あるいはコピー）してください。

```
Declare Function GetComputerName Lib "kernel32" _
    Alias "GetComputerNameA" (ByVal lpBuffer As String, _
    nSize As Long) As Long
```

このAPIは、引数「lpBuffer」に、コンピュータ名を格納する文字列型の変数を、「nSize」にこの変数のサイズ（バイト数）を設定して実行します。

5 フォームのコマンドボタンをクリックすると、ラベルコントロールにコンピュータ名を出力するようにするので、コマンドボタンのClickイベントプロシージャですべての処理を行います。

まず、コンピュータ名を格納する変数を、文字列型の文字数指定で宣言します。

```
Dim name As String * 16
```

変数のデータ型「As String」の後に、「*」と整数を記述すると、サイズを固定した変数を確保できます。API「GetComputerName」では、引数「nSize」に指定したバイト数と、コンピュータ名を格納する変数のサイズが同じでなければなりません。設定する数値は「入力可能なコンピュータ名の最大値（15バイト）＋1バイト」です。つまり変数を「16バイト」の固定サイズで確保します。

次に、API「GetComputerName」を実行します。宣言した変数「name」を第一引数に指定し、第二引数には変数で確保したサイズ「16」を記述します。

```
GetComputerName name, 16
```

6 これで、このAPIが実行されると、変数「name」に、コンピュータ名が格納されます。あとは、この名前（変数name）をラベルコントロールの［Caption］プロパティに設定すれば、コンピュータ名を表示できます。

```
Label1.Caption = name
```

Form1

使用しているコンピュータ名

EGG-POWER

表示されたコンピュータ名。使用しているコンピュータ名は、コントロールパネルの［ネットワーク］を開き、［ユーザー情報］タブで確認してみてください。

まとめ

簡単に、APIの概要と特徴、Visual Basicで使用する方法について説明しました。APIは直接オペレーティングシステムを操作するため、使いこなすにはWindowsの構造や基本的な動作原理に対する理解が必要になります。ぜひ、機会を捉えてこれらの勉強をすることをお勧めします。

また、どのような種類のAPIがあって、どういうときに使用するのかがわからないと使えません。Win32 SDKや『Windows95 APIバイブル 1』などを使って、これらの情報を入手してください。

練習問題

Q1　フォームにコマンドボタンをもう1つ追加し、APIを使ってWindowsがインストールされているディレクトリ名を、取得して表示するように修正してください。

【ヒント】

APIは、「GetWindowsDirectory」を使用します。Visual BasicでのDeclare宣言は以下のとおりです。

```
Declare Function GetWindowsDirectory Lib "kernel32" _
    Alias "GetWindowsDirectoryA" (ByVal lpBuffer As _
    String, ByVal nSize As Long) As Long
```

解答は巻末に→

Lesson 26

第８日
２時限目

Win32API を使う

Lesson26.vbp
VB6CD2
Lesson26

APIビューアを使う

Visual Basic 6.0の「Professional Edition」以上には、「APIビューア」というプログラムが付いています。このプログラムは、Win32APIをリスト表示し、Visual Basicで使用できるように、「Declare」ステートメントで宣言し直した記述を表示してくれる、大変優れたツールです。ここでは、このAPIビューアを使って、前のレッスンで作成した「コンピュータ名を取得するプログラム」にAPI関数を組み込みながら、APIとAPIビューアの使い方について理解を深めていきます。

今回作成するプログラム

Form1
現在のコンピュータ名
使用しているコンピュータ名
EGG-POWER
新しいコンピュータ名の設定
PCG-880 SERIES
コンピュータ名の設定
OK
キャンセル
コンピュータ名の設定
コンピュータ名を PCG-880 SERIES に設定しました。
この設定は再起動後に有効になります。
OK

ここをクリックすると、現在のコンピュータ名が表示される

新しいコンピュータ名を入力してこのボタンをクリックすると、名前を変更できる

再起動後に有効になることを知らせるメッセージが表示される

POINT

前レッスンで作成した、「コンピュータ名を取得するプログラム」に、今度はコンピュータ名を設定するAPI関数を組み込みます。現在のコンピュータ名を表示する機能と、新しくコンピュータ名を設定する機能を持ったプログラムに仕上げます。

コントロールとプロパティの設定

※新しいプロジェクトファイル（標準EXE）を用意します。

1 フォームにフレームコントロールを2つ配置し、各コントロールの次のプロパティを設定する

フレームコントロール	[Caption] プロパティ
Frame1	現在のコンピュータ名
Frame2	新しいコンピュータ名の設定

2 各フレーム内とフォームに2つずつコマンドボタンを配置し、プロパティとオブジェクト名を変更する

コントロール	[Caption] プロパティ	（オブジェクト名）
Command1	使用しているコンピュータ名	GetBtn
Command2	コンピュータ名の設定	SetBtn
Command3	OK	OkBtn
Command4	キャンセル	CxBtn

Form1
現在のコンピュータ名
使用しているコンピュータ名
新しいコンピュータ名の設定
コンピュータ名の設定
OK
キャンセル

配置してプロパティを設定したフレームコントロールとコマンドボタンコントロール

フレームを配置する

フレームコントロールを配置するには、ツールパレットの［Frame］を使います。

コマンドボタンを配置する

コマンドボタンを配置するには、ツールパレットの［CommandButton］を使います。

標準モジュールを挿入する

［プロジェクト］メニューの［標準モジュールの追加］を選択して、標準モジュールを挿入することもできます。

［標準モジュールの追加］ダイアログボックスが表示されるので、［標準モジュール］アイコンをクリックして、［開く］をクリックします。

3 プロジェクトに標準モジュールを追加する

1 ここをクリック
2 ここをクリック

4 ［スタート］→［プログラム］→［Microsoft Visual Basic 6.0］→［Microsoft Visual Basic 6.0 Tools］→［APIビューア］を選ぶ

1 ここをクリック

5 ［APIビューア］の［ファイル］メニューから［テキストファイルの読み込み］を選ぶ

1 ここをクリック
2 ここをクリック

6 「WIN32API.TXT」を選び、[開く] をクリックする

テキスト API ファイルを開く
ファイルの場所(I): Winapi
Apiload.txt
Mapi32.txt
Win32api.txt
ファイル名(N): Win32api.txt
ファイルの種類(T): テキスト ファイル (*.TXT)
開く(O)
キャンセル

1 ここをクリック

2 ここをクリック

7 データベースに変換するので、[ファイル] メニューから [データベースに変換] を選ぶ

APIビューア - C:¥Program Files¥Microsoft Visual Studio¥Common¥Tools¥W...
ファイル(F) 編集(E) 表示(V) ヘルプ(H)
テキスト ファイルの読み込み(T)...
データベース ファイルの読み込み(D)...
データベースに変換(V)...
C:¥Program Files¥Microsoft Visual Studio¥Common¥Tools¥Winapi¥Win32api.txt
APIビューアの終了(X)
AbortDoc
AbortPath
AbortPrinter
追加(A)
適用範囲

1 ここをクリック

8 「WIN32API.MDB」というファイル名で保存する

新規データベース名の選択
保存する場所(I): Winapi
ファイル名(N): Win32api.MDB
ファイルの種類(T): MDB ファイル (*.MDB)
保存(S)
キャンセル

1 ファイル名を入力

2 ここをクリック

9 [種類] から [関数] を選ぶ

APIビューア - C:¥Program Files¥Microsoft Visual Studio¥Common¥Tools¥W...
ファイル(F) 編集(E) 表示(V) ヘルプ(H)
種類(P):
関数
定数
関数
型
入力してください(Y):
有効な項目(I):
AbortDoc
AbortPath
追加(A)

1 ここをクリック

2 ここをクリック

WIN32API.TXT

通常、このファイルは「C:¥Program Files¥Microsoft Visual Studio¥Common¥Tools¥Winapi」フォルダの中にあります。

アラートが出ないときは

特にアラートの表示が出ないときは、[ファイル] メニューから [データベースに変換] を選んでください。

関数名しか表示されない

［選択された項目］の欄に関数名しか表示されない場合は、［表示］メニューの［すべてのテキスト］を選びます。

素早く項目を探すには

リストから素早く目的の項目探し出すには、関数名の最初の何文字かをキーボードでタイプします。例えば、「SetComputer Name」なら、「S」「e」「t」と素早くタイプしてみてください。すぐに「Set～」の項目が選択表示になります。

10 ［有効な項目］から［SetComputerName］を選び、［追加］ボタンをクリックする

1 ここをクリック

2 ここをクリック

［選択された項目］に「Declare」ステートメントによる宣言コードが表示される

11 今度は［有効な項目］から［GetComputerName］を選び、［追加］ボタンをクリックする

1 ここをクリック

2 ここをクリック

［選択された項目］に宣言コードが追加される

12 ［コピー］ボタンをクリックする

1 ここをクリック

13 Visual Basicに戻り、標準モジュールにカーソルを置き、[貼り付け] ボタンをクリックする

Lesson26 - Module1 (コード)
(General)　(Declarations)

1 標準モジュールにカーソルを置く

Lesson26 - Microsoft Visual Basic [デザイン]
ファイル(F) 編集(E) 表示(V) プロジェクト(P) 書式(O) デバッグ
標準
貼り付け

2 ここをクリック

Lesson26 - Module1 (コード)
(General)　(Declarations)

```
Public Declare Function SetComputerName Lib "kernel32" Alias
Public Declare Function GetComputerName Lib "kernel32" Alias
```

標準モジュールにAPIの宣言コードが組み込まれる

14 ラベルコントロールを上のフレームコントロール内に配置し、次のプロパティを設定する

プロパティ	プロパティ値
Alignment	2 ー中央揃え
BorderStyle	1 ー実線
Caption	(なし)

Form1
現在のコンピュータ名
使用しているコンピュータ名
新しいコンピュータ名の設定

配置してプロパティを設定したラベルコントロール

ラベルコントロールを配置する

ラベルコントロールを配置するには、ツールパレットの［Label］ A を使います。

テキストボックスを配置する

テキストボックスコントロールを配置するには、ツールパレットの[TextBox] abl を使います。

15 テキストボックスを下のフレームコントロール内に配置し、次のプロパティを設定する

プロパティ	プロパティ値
Text	(なし)
TabIndex	0

Form1

現在のコンピュータ名

使用しているコンピュータ名

新しいコンピュータ名の設定

コンピュータ名の設定

OK キャンセル

配置してプロパティを設定したテキストボックスコントロール

コードの記述

Module1:(General)-(Declarations)

Lesson26 - Module1 (コード)

(General) (Declarations)

```
Public Declare Function SetComputerName Lib "kernel32" _
       Alias "SetComputerNameA" _
       (ByVal lpComputerName As String) As Long

Public Declare Function GetComputerName Lib "kernel32" _
       Alias "GetComputerNameA" _
       (ByVal lpBuffer As String, nSize As Long) As Long
```

Form1:(General)-(Declarations)

Lesson26 - Form1 (コード)

(General) | (Declarations)

```
Dim CompName As String * 20
```

Form1:Form-Load

Lesson26 - Form1 (コード)

Form | Load

```
Private Sub Form_Load()
    GetComputerName CompName, 20
End Sub
```

Form1:GetBtn-Click

Lesson26 - Form1 (コード)

GetBtn | Click

```
Private Sub GetBtn_Click()
    GetComputerName CompName, 20
    Label1.Caption = CompName
End Sub
```

Form1:SetBtn-Click

Lesson26 - Form1 (コード)

SetBtn | Click

```
Private Sub SetBtn_Click()
    Dim SetName As String
    Dim Ret As Long

    SetName = Text1.Text
    If SetName = "" Then
        MsgBox "設定する名前を入力してください。"
        Text1.SetFocus
        Exit Sub
    Else
        Ret = SetComputerName(SetName)
        If Ret = False Then
            MsgBox "文字数が制限を越えています。再度入力し直してください。"
            Text1.SetFocus
            Exit Sub
        End If
    End If

    Text1.SetFocus
```

```
        MsgBox "コンピュータ名を " & SetName & " に設定しました。" _
              & Chr(13) & Chr(13) & _
              "この設定は再起動後に有効になります。", vbOKOnly, _
              "コンピュータ名の設定"
End Sub
```

Form1:OkBtn-Click

Lesson26 - Form1 (ｺｰﾄﾞ)

OkBtn | Click

```
Private Sub OkBtn_Click()
    Unload Form1
    End
End Sub
```

Form1:CxBtn-Click

Lesson26 - Form1 (ｺｰﾄﾞ)

CxBtn | Click

```
Private Sub CxBtn_Click()
    SetComputerName CompName
    Unload Form1
    End
End Sub
```

解 説

1

Win32APIには、「Get」ではじまる名前の関数が大変多くあります。これらのAPI関数は、主にいろいろな情報をWindowsから取得してくる関数です。一方、「Set」ではじまる関数もかなりの数が用意されています。これらは、逆にWindowsに対して情報を設定するAPI関数になっています。

APIによっては、「Get」と「Set」の名前が同じものがあります。これらは、同じ種類の情報を取り出し、また設定する関数です。「GetComputerName」も、同じ名前で対になった「SetComputerName」というAPIがあります。このAPI関数は、その名のとおりWindowsに対してコンピュータ名を設定するAPIになっています。

本プログラムでは、API関数「SetComputerName」を使って、新しいコンピュータ名をWindowsに設定するプログラムを作成します。

2

API関数「SetComputerName」は、引数に設定する文字列を指定して実行すると、そのままコンピュータ名を設定できます。ただし、設定した新しいコンピュータ名は、Windowsを再起動した後に有効になります。

このプログラムでは、テキストボックスコントロールに文字を入力してもらい、

［コンピュータ名の設定］ボタンをクリックした時点で入力されたコンピュータ名を、「SetComputerName」でWindowsに設定します。

新しいコンピュータ名の設定
SETO VAIO
コンピュータ名の設定
ここにコンピュータ名を入力し…
ここをクリックして設定する

ネットワーク
ネットワークの設定　識別情報　アクセスの制御
次の情報は、ネットワーク上でコンピュータを識別するために使われます。 このコンピュータの名前と所属するワークグループ名、簡単な説明を入力してください。
コンピュータ名：　SETO VAIO
ワークグループ：　Workgroup
コンピュータの説明：
コンピュータ名が変更された

3　前のレッスンでは、APIの「Declare」ステートメントによる宣言を、直接手で入力しました。しかし、Visual Basicのプロフェッショナルエディションには、「APIビューア」という便利なツールが付属しています。すでに画面の操作解説のところで、「APIビューア」の操作方法を紹介しました。このツールを使えば、難解なAPIの再定義が簡単に行えます。このレッスンでは、「SetComputerName」と「GetComputerName」の2つの関数の「Declare」宣言を探し出し、コードに貼り付けています。

「APIビューア」は、APIの「関数」「定数」「型」を、Visual Basicで使用できる形式に自動的に変換してくれます。使用したいこれらの名前がわかれば、「APIビューア」を使用することで、Visual Basicでの再宣言作業がとても楽になります。たとえば、「C言語」で使用される「構造体」を引数に持つAPI関数では、その構造体の「型」をリストで探し出せば、Visual Basicのユーザー定義のデータ型に変換して表示してくれます。

ただし、「APIビューア」はあくまでも関数名とその宣言文を表示するだけですから、どの関数がどういう機能を持っているのかは、『Win32 SDK』や『Windows95 APIバイブル 1』などを使わないとわかりません。

APIビューアの関数リスト

定数リスト

型リスト

4 では、実際にコードを作成しましょう。標準モジュールには、すでに「Declare」ステートメントによるAPI関数「SetComputerName」と「GetComputerName」が再宣言されています（Declareステートメントの書式はP.350を参照）。

```
Declare Function SetComputerName Lib "kernel32" _
    Alias "SetComputerNameA" _
    (ByVal lpComputerName As String) As Long

Declare Function GetComputerName Lib "kernel32" _
    Alias "GetComputerNameA" _
    (ByVal lpBuffer As String, nSize As Long) As Long
```

「SetComputerName」関数は、引数「lpBuffer」に新しく設定するコンピュータ名を、文字列またはString型の変数で指定します。コンピュータ名として設定できる最大文字数は、Windows95/98、Windows NT 4.0ともに半角換算で15文字までです。スペース以外の標準のASCII文字と数字、そして次の記号が使用できます。

```
! @ # $ % ^ & ' ) ( . - _ { } ~ .
```

「SetComputerName」関数は、成功すると「True」を、失敗すると「False」を返してきます。

5

標準モジュールでAPI関数の宣言をしたら、フォームのコードに実際の処理を作成します。

まず、宣言セクションに変数を宣言します。「CompName」は、現在設定されているコンピュータ名を格納します。余裕を見て20バイトの固定長で宣言します。

```
Dim CompName As String * 20
```

そして、フォームのLoadイベントプロシージャで、API関数「GetComputerName」を実行します。プログラムの［キャンセル］ボタンがクリックされた場合は、入力されたコンピュータ名を取り消して、元の設定に戻しておくためです。

```
Private Sub Form_Load()
    GetComputerName CompName, 20
End Sub
```

6

フォームに配置した［使用しているコンピュータ名］ボタンをクリックすると、前のレッスンで行ったように、Windowsからコンピュータ名を取得して、ラベルコントロールに表示します。すでにプログラム起動時に、変数に格納しているので、この処理は必要ないのですが、前レッスンの復習の意味で付けておきました。

```
Private Sub GetBtn_Click()
    GetComputerName CompName, 20
    Label1.Caption = CompName
End Sub
```

7

今度は、［コンピュータ名の設定］ボタンがクリックされたときの処理を作成します。このイベントプロシージャで、新しいコンピュータ名の設定処理を行います。

まず、変数「SetName」を宣言します。この変数は、設定するコンピュータ名を格納する変数で、文字列型で宣言します。変数「Ret」は、API「SetComputerName」関数が返してくる戻り値を格納する変数です。このAPI関数は、Long型で宣言されていますから、同じデータ型で宣言しておきます。

```
Private Sub SetBtn_Click()
    Dim SetName As String
    Dim Ret As Long
```

次に、テキストボックスコントロールに入力された文字列を、変数に格納します。

```
SetName = Text1.Text
```

テキストボックスに何も入力されていないのに、このボタンがクリックされて「SetComputerName」関数が実行されると、プログラムはエラーになってしまいます。そこで、変数「SetName」の値が""（空白）であれば、メッセージボックスを表示し、「Exit Sub」ステートメントでプロシージャを終了させます。このとき、直前にテキストボックスにフォーカスを移して、次の入力に備えます。

```
If SetName = "" Then
    MsgBox "設定する名前を入力してください。"
    Text1.SetFocus
    Exit Sub
```


8 文字列が入力されていれば、「SetComputerName」関数を実行します。このとき、関数の戻り値を変数「Ret」に格納しておきます。

入力された文字の文字数が、設定できる最大文字数（半角換算で15文字）を超えていると、「SetComputerName」関数は新しいコンピュータ名を設定できずに「False」を返してきます。そこで、「False」が返ってきた場合は、再度入力をやり直してもらうように、メッセージボックスを表示し、プロシージャを終了します。

```
Else
    Ret = SetComputerName(SetName)
    If Ret = False Then
        MsgBox "文字数が制限を越えています。再度入力し直してください。"
        Text1.SetFocus
        Exit Sub
    End If
End If
```

再入力を促すメッセージボックスが表示される

9　入力された文字が、正しく設定できた場合は、設定したコンピュータ名をメッセージボックスで表示します。「SetComputerName」関数は、あくまでもコンピュータ名を設定するだけです。実際に設定されるには、コンピュータを再起動する必要があります。そこで、メッセージボックスにこの内容も表示するようにします。

```
Text1.SetFocus
MsgBox "コンピュータ名を " & SetName & " に設定しました。" _
    & Chr(13) & Chr(13) & _
    "この設定は再起動後に有効になります。", vbOKOnly, _
    "コンピュータ名の設定"
```

再起動後に有効になることを知らせるメッセージ

10　最後に、プログラムの終了処理を作成します。まず、［OK］ボタンがクリックされれば、そのままプログラムの終了処理を実行します。「SetComputerName」関数による新しいコンピュータ名の設定は有効のまま、プログラムが終了します。

```
Private Sub OkBtn_Click()
    Unload Form1
    End
End Sub
```

一方、[キャンセル] ボタンがクリックされた場合は、新しいコンピュータ名の設定を取り消して、プログラムが起動する前に戻しておく必要があります。そこで、プログラム起動時に格納していた、現在のコンピュータ名を使って、再度「SetComputerName」関数を実行します。これで、一度別のコンピュータ名に設定を変更していても、元に戻すことが可能になります。そして、プログラムの終了処理を実行します。

```
Private Sub CxBtn_Click()
    SetComputerName CompName
    Unload Form1
    End
End Sub
```

Declareステートメントの書式

```
[Public | Private] Declare [Sub | Function] name
    Lib "libname" [Alias "aliasname"] [([arglist])]
```

Public/Private	省略可能。プロシージャの種類を宣言するときに指定します。
Sub/Function	Function か Sub のどちらかを指定します。Sub はプロシージャが値を返し、Function はプロシージャが値を返さないことを示します。
name	プロシージャ名を指定します。大文字小文字を区別して指定します。
Lib	必ず指定します。
libname	DLL のファイル名を指定します。
Alias	DLL の中で別の名前を持っている場合指定します。省略可能。
aliasname	DLL の中の別のプロシージャ名を指定します。省略可能。
arglist	プロシージャに渡す引数を表す変数のリストを指定します。省略可能。

ワンポイントアドバイス

APIビューアの操作解説で、対象となるテキストファイルを、データベースファイルに変換しました。この変換を行った場合は、次回APIビューアを使用するときに「MDB」ファイルを読み込んで使用すると、関数などの検索が高速になります。

操作は、[ファイル] メニューの [データベースファイルの読み込み] を選択して、「WIN32API.MDB」を開くだけです。ちなみに、このファイルは、Microsoft Accessのデータベースファイルになっています。

まとめ

「API ビューア」を使うと、API 関数の再定義がとても簡単にできます。おそらく、これらのツールがないと、Visual Basic に API を組み込むために、C 言語の知識も必要になり、大変な作業でしょう。

プログラムでは、Windows から情報を取得する API 関数と、設定する API 関数の 2 つを組み合わせて使用しました。

設定を変更する場合は、事前に変更前の情報をきちんと保存しておいて、いつでも元に戻せるようにしておきましょう。間違った設定によって Windows が再起動しなくなってしまう、ということは良くあることです。API は便利ですが、使う場合は、十分注意して実装してください。

練習問題

Q1　作成したプログラムに、フォームをクリックするとフォームのタイトル（ウィンドウタイトル）が点滅する機能を組み込んでください。また、点滅を中止する［リセット］ボタンも組み込んでみましょう。

【ヒント】

API 関数「FlashWindow」を使用します。Declare ステートメントによる再宣言の記述は、次のとおりです。API ビューアをお持ちの方は、操作方法の練習を兼ねて、この関数を標準モジュールに組み込んでください。

```
Declare Function FlashWindow Lib "user32" _
   (ByVal hwnd As Long, _
   ByVal bInvert As Long) As Long
```

解答は巻末に→

Lesson 27

第８日
３時限目

Win32API
を使う

Lesson27.vbp
VB6CD2
Lesson27

メモリチェッカーの作成

これまでのレッスンで、Win32APIの使い方とAPIビューアの操作方法を覚えました。使用したAPI関数は、比較的簡単な使い方の関数でした。このレッスンでは、もう少し複雑なデータ構造を引数に持つAPI関数を使用して、コンピュータが現在使用しているメモリの状況を表示するプログラムを作成します。

今回作成するプログラム

POINT

API関数「GlobalMemoryStatus」の使い方と、返してきた値をコントロールに反映させるテクニックがポイントです。この関数は、引数にC言語で言う「構造体」を持ちます。Visual Basicでは、「ユーザー定義型」と呼ばれるデータ型を使用して、メモリに関する情報を操作します。

コントロールとプロパティの設定

※新しいプロジェクトファイル（標準EXE）を用意します。

1 フォームの［Caption］プロパティを「Memory Checker」に変更する

(ｵﾌﾞｼﾞｪｸﾄ名)	Form1
Appearance	1 - 3D
AutoRedraw	False
BackColor	&H8000000F&
BorderStyle	2 - 可変
Caption	Memory Checker
ClipControls	True
ControlBox	True

1 ここをクリック

2 コマンドボタンコントロールを配置し、［Caption］プロパティを「Memory Check」に変更する

Memory Checker

Memory Check

配置してプロパティを変更したコマンドボタンコントロール

コマンドボタンを配置する

コマンドボタンを配置するには、ツールパレットの［CommandButton］を使います。

3 ラベルコントロールを4つ配置し、各コントロールの［Caption］プロパティを変更する

ラベルコントロール	[Caption] プロパティ
Label1	総メモリ
Label2	使用可能なメモリ
Label3	総仮想メモリ
Label4	使用可能な仮想メモリ

ラベルコントロールを配置する

ラベルコントロールを配置するには、ツールパレットの［Label］A を使います。

配置してプロパティを設定したラベルコントロール

4 ラベルコントロールをもう１つ配置して次のプロパティを設定し、コピーを３つ作成する

プロパティ	設定
Alignment	１－右揃え
BorderStyle	１－実線
Caption	（なし）

配置してプロパティを設定したラベルコントロール

コピーして配置したラベルコントロール

ラベルコントロールをコピーする

配置してプロパティを設定したラベルコントロールをクリックして選択し、［標準］ツールバーの［コピー］ボタンをクリックします。続けて［貼り付け］ボタンをクリックし、作成されたラベルコントロールを適切な位置までドラッグして配置します。

5 プロジェクトに標準モジュールを挿入する

1 ここをクリック

2 ここをクリック

6 APIビューアを起動し、関数［GlobalMemoryStatus］を選び［追加］ボタンをクリックする

1 ［関数］を選択

2 ここをクリック

3 ここをクリック

「Declare」ステートメントによる宣言コードが表示される

7 ［種類］から［型］を選び、［有効な項目］から［MEMORYSTATUS］を選ぶ

1 ［型］を選択

2 ここをクリック

標準モジュールを挿入する

［プロジェクト］メニューの［標準モジュールの追加］を選択して、標準モジュールを挿入することもできます。

［標準モジュールの追加］ダイアログボックスが表示されるので、［標準モジュール］アイコンをクリックして、［開く］をクリックします。

APIビューアを起動する

APIビューアを起動するには、タスクバーの［スタート］ボタン→［プログラム］→［Microsoft Visual Basic 6.0］→［Microsoft Visual Basic 6.0 Tools］→［APIビューア］を選択します。

8 ［追加］ボタンをクリックし、［コピー］ボタンをクリックする

```
API ビューア - C:¥Program Files¥Microsoft Visual Studio¥Common¥Tools¥W...
ファイル(F) 編集(E) 表示(V) ヘルプ(H)
種類(P):
型
検索する文字列の先頭の文字をいくつか入力してください(Y):

有効な項目(I):
MEASUREITEMSTRUCT
MEMORY_BASIC_INFORMATION
MEMORYSTATUS
MENU_EVENT_RECORD
MENUITEMINFO
MENUITEMTEMPLATE
MENUITEMTEMPLATEHEADER
METAFILEPICT
選択された項目(S):
Public Declare Sub GlobalMemoryStatus Lib "kernel32"
Alias "GlobalMemoryStatus" (lpBuffer As
MEMORYSTATUS)

Public Type MEMORYSTATUS
        dwLength As Long
        dwMemoryLoad As Long
        dwTotalPhys As Long
        dwAvailPhys As Long
        dwTotalPageFile As Long
        dwAvailPageFile As Long
リスト ボックス内に直接入力した文字で検索されます。
```

追加(A)　適用範囲　Public(U)　Private(T)　削除(R)　クリア(L)　コピー(C)

1 ここをクリック

ユーザー定義のデータ型が表示される

2 ここをクリック

9 Visual Basicに戻り、標準モジュールにカーソルを置き、［貼り付け］ボタンをクリックする

Lesson27 - Module1 (コード)

(General)　(Declarations)

```
Public Declare Sub GlobalMemoryStatus Lib "kernel32" _
            (lpBuffer As MEMORYSTATUS)

Type MEMORYSTATUS
        dwLength As Long
        dwMemoryLoad As Long
        dwTotalPhys As Long
        dwAvailPhys As Long
        dwTotalPageFile As Long
        dwAvailPageFile As Long
        dwTotalVirtual As Long
        dwAvailVirtual As Long
End Type
```

標準モジュールにAPIのコードが組み込まれる

コードの記述

Module1:(General)-(Declarations)

Lesson27 - Module1 (コード)

(General) / (Declarations)

```
Declare Sub GlobalMemoryStatus Lib "kernel32" _
                    (lpBuffer As MEMORYSTATUS)

Type MEMORYSTATUS
        dwLength As Long
        dwMemoryLoad As Long
        dwTotalPhys As Long
        dwAvailPhys As Long
        dwTotalPageFile As Long
        dwAvailPageFile As Long
        dwTotalVirtual As Long
        dwAvailVirtual As Long
End Type
```

Form1:Command1-Click

Lesson27 - Form1 (コード)

Command1 / Click

```
Private Sub Command1_Click()
    Dim Memdata As MEMORYSTATUS

    GlobalMemoryStatus Memdata

    With Memdata
        Label5(0).Caption = .dwTotalPhys
        Label5(1).Caption = .dwAvailPhys
        Label5(2).Caption = .dwTotalPageFile
        Label5(3).Caption = .dwAvailPageFile
    End With
End Sub
```

解説

1 このプログラムは、コマンドボタン［Memory Check］をクリックすると、コンピュータで使用している現在のメモリと仮想メモリの総容量、使用可能な容量を表示するプログラムです。

このメモリに関する情報は、Visual Basicのコントロールや関数などでは取得できません。これらは、API関数「GlobalMemoryStatus」を使えば把握できます。

Memory Checker

Memory Check ← ここをクリックすると…

総メモリ 66084864
使用可能なメモリ 2256896
総仮想メモリ 1881571328
使用可能な仮想メモリ 1815859200

コンピュータのメモリ容量を表示

2 API関数「GlobalMemoryStatus」は、実行するとメモリに関する8つの情報をWindowsから把握し、引数に設定した「構造体」（ユーザー定義のデータ型）に格納します。

```
Declare Sub GlobalMemoryStatus Lib "kernel32" _
    (lpBuffer As MEMORYSTATUS)
```

「MEMORYSTATUS」と定義されているのが、そのユーザー定義のデータ型で、このAPI関数を実装する場合は、「MEMORYSTATUS」というデータ型を定義しなくてはなりません。その場合は、APIビューアの「型」リストを探せば、再定義に必要な記述が表示されます。

ファイル(F) 編集(E) 表示(V) ヘルプ(H)
種類(P):
型
検索する文字列の先頭の文字をいくつか入力してください(W):
有効な項目(I):
MEASUREITEMSTRUCT
MEMORY_BASIC_INFORMATION
MEMORYSTATUS
MENU_EVENT_RECORD
MENUITEMINFO
MENUITEMTEMPLATE
MENUITEMTEMPLATEHEADER
METAFILEPICT
追加(A)
適用範囲
Public(U)
Private(T)
選択された項目(S):

```
Type MEMORYSTATUS
        dwLength As Long
        dwMemoryLoad As Long
        dwTotalPhys As Long
        dwAvailPhys As Long
        dwTotalPageFile As Long
        dwAvailPageFile As Long
        dwTotalVirtual As Long
        dwAvailVirtual As Long
End Type
```

削除(R)
クリア(L)
コピー(C)

型リストで［MEMORYSTATUS］を選択すると…

再定義に必要な記述が表示される

3　「構造体（ユーザー定義のデータ型）」とは、複数の変数を一括で管理できるように用意されているデータ型です。「文字列型（String）」とか「整数型（Integer）」などのように、あらかじめVisual Basicで定められているデータ型ではなく、ユーザーが既存のデータ型を組み合わせて独自のデータ型を作成できるようになっています。この「ユーザー定義のデータ型」を定義する場合は、必ず「Type」ステートメントで型の名前を宣言し、組み込む変数を列挙します。「Type」ステートメントの中は、Dimステートメントの記述は必要なく、そのまま変数名とデータ型を指定します。ここでは、8つのLong型変数を、ユーザー定義のデータ型の要素として宣言しています。

```
Type MEMORYSTATUS
    dwLength As Long
    dwMemoryLoad As Long
    dwTotalPhys As Long
    dwAvailPhys As Long
    dwTotalPageFile As Long
    dwAvailPageFile As Long
    dwTotalVirtual As Long
    dwAvailVirtual As Long
End Type
```

ユーザー定義のデータ型の名前は、通常の変数の名前付き規則に従います。また、各要素の変数のデータ型は、同じ型でなくてもかまいません。次のユーザー定義型「MYFRIEND」は、いろいろなデータ型の変数を要素としています。

```
Type MYFRIEND
    Name As String
    Age As Integer
    PhoneNo As String
    BirthDay As Date
End Type
```

ユーザー定義のデータ型は、必ず「Type」ステートメントで始まり、「End Type」で終了してください。

4　このプログラムは、コマンドボタンをクリックすると、Windowsから現在のメモリの使用状況を把握し、変数に格納します。そこで、コマンドボタン［Memory Check］のClickイベントプロシージャに、その処理を作成します。

まず、標準モジュールで定義したユーザー定義のデータ型「MEMORYSTATUS」を、コードで使用できるように、Dimステートメントで変数として宣言します。このときに、定義したユーザー定義のデータ型を指定します。ここでは、「Memdata」という変数を、定義した「MEMORYSTATUS」で型指定しています。

```
Private Sub Command1_Click()
    Dim Memdata As MEMORYSTATUS
```

5 ユーザー定義のデータ型で変数を宣言したら、この変数を引数にして、API関数「GlobalMemoryStatus」を実行します。

```
GlobalMemoryStatus Memdata
```

実行した結果は、変数「Memdata」の各要素に格納されます。変数の各要素にアクセスするには、変数名の後に「.」を付けて、ユーザー定義のデータ型「MEMORYSTATUS」を定義したときの、各要素の変数名を指定します。たとえば、変数「dwTotalPhys」の値を把握したければ、

```
Memdata.dwTotalPhys
```

と記述します。この方法で、4つの変数の値を、ラベルコントロールの［Caption］プロパティに出力します。

```
With Memdata
    Label5(0).Caption = .dwTotalPhys
    Label5(1).Caption = .dwAvailPhys
    Label5(2).Caption = .dwTotalPageFile
    Label5(3).Caption = .dwAvailPageFile
End With
```

ワンポイントアドバイス

「GlobalMemoryStatus」は、どのデータをどの変数に格納するのかがあらかじめ決まっており、その規則に合ったユーザー定義のデータ型を宣言しておかなければなりません。そのために、データ型の定義までもAPIビューアを使って、API関数の宣言で指定したデータ型と一致するデータ型を定義しています。

6 これで完成です。プログラムを起動して、［Memory Check］ボタンをクリックすると、現在のメモリの使用状況が表示されます。

他のアプリケーションを起動したり終了したのちに、プログラムの［Memory Check］ボタンをクリックすると、値は変化し常に最新のメモリ状況を知ることができます。

ワンポイントアドバイス

ユーザー定義のデータ型「MEMORYSTATUS」の、各要素の役割は次のとおりです。

名称	機能
dwLength As Long	このデータ型のサイズ（長整数型）
dwMemoryLoad As Long	メモリの使用状況のパーセント（長整数型）
dwTotalPhys As Long	物理メモリの総バイト数（長整数型）
dwAvailPhys As Long	使用可能な物理メモリのバイト数（長整数型）
dwTotalPageFile As Long	ページファイルに格納できる総バイト数（長整数型）
dwAvailPageFile As Long	使用可能なページファイル内のバイト数（長整数型）
dwTotalVirtual As Long	呼び出しもとのプロセスで使用できる仮想メモリの総バイト数（長整数型）
dwAvailVirtual As Long	使用可能な仮想メモリのバイト数（長整数型）

なお、「物理メモリ」とは実際にコンピュータに取り付けているメモリのことです。また、コンピュータが使用できる仮想メモリの量や使用量を知るには、「dwTotalPageFile」と「dwAvailPageFile」を使用してください。Windows98やWindows NT 4.0では、仮想メモリを「ページングファイル」として設定しています。これらは、コントロールパネルの「システム」アプレットで確認できます。また、Windows95/98では、仮想メモリを自動設定にしている場合は、ハードディスクの空き容量に依存します。

Windows98の「システムのプロパティ」

まとめ

API関数を使用する場合は、引数に定められているデータ型と一致したデータ型を使用しないと、関数の実行が失敗します。定数や型も、APIビューアで調べることができますから、この機能を使って定義をしてください。

定義したユーザー定義のデータ型を使用する場合は、使用するコードでDimステートメントを使って変数宣言をする際に、データ型指定をすれば使用できます。複数のデータを一元管理する場合にも使用できますので、活用してください。

練習問題

Q1 作成したプログラムに、「使用済みのメモリ量」を表示するように修正してください。このとき、「40MB」などと「メガバイト」で表示するようにしてみましょう。

Memory Checker
Memory Check
総メモリ 66MB
使用済みメモリ 56MB
使用可能なメモリ 10MB
総仮想メモリ 1882MB
使用可能な仮想メモリ 1826MB
追加した項目

解答は巻末に→

Let's VBSchool!!

第９日 (Lesson28～Lesson31)
APIを使ったアプリケーションの作成

これまでのレッスンで、Visual Basicのプログラムに Win32APIを組み込む方法を紹介しました。今日のレッスンでは、さらにいろいろなAPIを組み込んで、Visual Basicだけでは実現できないアプリケーションを作成します。

Lesson28の「バージョン情報表示プログラム」は、Windowsのバージョン情報を表示します。Lesson29～Lesson30では、デスクトップの壁紙を自由に変更できるプログラムを２つ作成します。Lesson31では、指定したディスクの使用量や空き容量を、グラフで表示するプログラムを作成します。いずれも、API関数を呼び出して組み込んでいます。

今日のレッスンで解説する項目

- ❏ API関数「GetVersionEx」の実行
- ❏ API関数「GetSystemInfo」の実行
- ❏ ユーザー定義型変数のサイズの取得
- ❏ Windowsのバージョン情報の概要
- ❏ API関数「SystemParametersInfo」の機能と使用方法
- ❏ デスクトップの壁紙の変更方法
- ❏ プログラム「壁紙チェンジャー」の作成
- ❏ プログラムを時刻によって自動実行する方法
- ❏ クラスタとセクタ
- ❏ ドライブの種類（タイプ）
- ❏ ディスク容量の把握
- ❏ CInt関数によるデータの整数変換操作
- ❏ グラフコントロールの使用方法

Lesson 28

第９日
１時限目

APIを使ったアプリケーションの作成

Lesson28.vbp
VB6CD2
Lesson28

バージョン情報表示プログラム

Win32APIには、オペレーティングシステムのバージョンに関する情報やコンピュータのCPUに関する情報を、Windowsから取り出す関数があります。このレッスンでは、これらの関数を使ったプログラムを作成します。

今回作成するプログラム

System Information
System Information
Windows Version
バージョン 4.10
BUILD : 67766222
Windows98
CPU Information
CPU Type: Pentium, 1 Processor

ここをクリックすると...
Windowsのバージョン番号情報と...
CPUの種類と個数が表示される

POINT

「GetVersionEx」と「GetSystemInfo」の２つのAPI関数を使用して、WindowsのバージョンとコンピュータのCPUに関する情報を取得し、フォームに表示するプログラムです。ポイントは、関数に渡す構造体（ユーザー定義のデータ型）の使い方にあります。

コントロールとプロパティの設定

※新しいプロジェクトファイル（標準EXE）を用意します。

1 フォームの［Caption］プロパティを「System Information」に変更する

2 コマンドボタンコントロールを配置し、［Caption］プロパティを「System Information」に変更する

3 フレームコントロールを2つ配置し、各コントロールの［Caption］プロパティを変更する

フレームコントロール	[Caption] プロパティ
Frame1	Windows Version
Frame2	CPU Information

配置してプロパティを変更したコマンドボタンコントロールとフレームコントロール

コマンドボタンを配置する

コマンドボタンを配置するには、ツールパレットの［CommandButton］を使います。

フレームコントロールを配置する

フレームコントロールを配置するには、ツールパレットの［Frame］を使います。

ラベルコントロールを配置する

ラベルコントロールを配置するには、ツールパレットの［Label］A を使います。

コントロールをコピーする

プロパティの設定が同じコントロールを複数作成する場合は、コントロールを1つ作成したら、それをコピーして貼り付けると効率的に作業できます。

標準モジュールを挿入する

［プロジェクト］メニューの［標準モジュールの追加］を選択して、標準モジュールを挿入することもできます。

［標準モジュールの追加］ダイアログボックスが表示されるので、［標準モジュール］アイコンをクリックして、［開く］をクリックします。

4 各フレーム内にラベルコントロールを配置し、各ラベルコントロールの次のプロパティを設定する

プロパティ	プロパティの値
Alignment	2－中央揃え
BorderStyle	1－実線
Caption	（なし）

5 プロジェクトに標準モジュールを挿入する

6 APIビューアを起動し、次の関数と型をコピーし、標準モジュールに貼り付ける

種類	有効な項目
[関数]	GetVersionEx GetSystemInfo
[型]	OSVERSIONINFO SYSTEM_INFO

API ﾋﾞｭｰｱ - C:¥Program Files¥Microsoft Visual Studio¥Common¥Tools¥W...
ﾌｧｲﾙ(F)　編集(E)　表示(V)　ﾍﾙﾌﾟ(H)
種類(P):
型
検索する文字列の先頭の文字をいくつか入力してください(Y):
sys
有効な項目(I):
SYSTEM_ALARM_ACE
SYSTEM_AUDIT_ACE
SYSTEM_INFO
SYSTEM_POWER_STATUS
SYSTEMTIME
Target
TEXTMETRIC
TIME_ZONE_INFORMATION
追加(A)
適用範囲
Public(U)
Private(T)
選択された項目(S):
Public Declare Function GetVersionEx Lib "kernel32" Alias "GetVersionExA" (lpVersionInformation As OSVERSIONINFO) As Long

Public Declare Sub GetSystemInfo Lib "kernel32" Alias "GetSystemInfo" (lpSystemInfo As SYSTEM_INFO)

Public Type OSVERSIONINFO
dwOSVersionInfoSize As Long
dwMajorVersion As Long
dwMinorVersion As Long
削除(R)
ｸﾘｱ(L)
ｺﾋﾟｰ(C)
ﾘｽﾄ ﾎﾞｯｸｽ内に直接入力した文字で検索されます。

1 ここで［関数］、［型］を選択

2 ［有効な項目］を選択し、［追加］をクリック

3 ここをクリック

APIビューアを起動する

APIビューアを起動するには、タスクバーの［スタート］ボタン→［プログラム］→［Microsoft Visual Basic 6.0］→［Microsoft Visual Basic 6.0 Tools］→［APIビューア］を選択します。

APIのコードをコピーする

APIのコードを標準モジュールにコピーする詳しい方法については、「Lesson 26 APIビューアを使う」を参照してください。

コードの記述

Module1:(General)-(Declarations)

Lesson28 - Module1 (ｺｰﾄﾞ)
(General)　(Declarations)

```
Public Declare Function GetVersionEx Lib "kernel32" _
    Alias "GetVersionExA" _
    (lpVersionInformation As OSVERSIONINFO) As Long

Type OSVERSIONINFO
        dwOSVersionInfoSize As Long
        dwMajorVersion As Long
        dwMinorVersion As Long
        dwBuildNumber As Long
```

```
        dwPlatformId As Long
        szCSDVersion As String * 128
End Type

Public Declare Sub GetSystemInfo Lib "kernel32" _
        (lpSystemInfo As SYSTEM_INFO)

Type SYSTEM_INFO
        dwOemID As Long
        dwPageSize As Long
        lpMinimumApplicationAddress As Long
        lpMaximumApplicationAddress As Long
        dwActiveProcessorMask As Long
        dwNumberOrfProcessors As Long
        dwProcessorType As Long
        dwAllocationGranularity As Long
        dwReserved As Long
End Type
```

Form1:Command1-Click

Lesson28 - Form1 (ｺｰﾄﾞ)

Command1 | Click

```
Private Sub Command1_Click()
    Dim Msg1 As String, Msg2 As String
    Dim Info(2) As String
    Dim SysInfo As SYSTEM_INFO
    Dim VInfo As OSVERSIONINFO

    VInfo.dwOSVersionInfoSize = Len(VInfo)
    GetVersionEx VInfo
    With VInfo
        Msg1 = "バージョン " & .dwMajorVersion & "." _
            & .dwMinorVersion
        Label1(0).Caption = Msg1
        Label1(1).Caption = "BUILD : " & .dwBuildNumber
        Select Case .dwPlatformId
            Case 0
                Msg2 = "Win32s on Windows3.1"
            Case 1
                If .dwMinorVersion > 0 Then
                    Msg2 = "Windows98"
                Else
                    Msg2 = "Windows95"
                End If
            Case 2
                Msg2 = "WindowsNT"
```

```
        End Select
        Label1(2).Caption = Msg2
        Label1(3).Caption = .szCSDVersion
    End With

    GetSystemInfo SysInfo

    With SysInfo
        Info(0) = .dwNumberOrfProcessors & " Processor"
        Select Case SysInfo.dwProcessorType
            Case 586
                Info(1) = "Pentium"
        End Select

        Info(2) = "CPU Type: " & Info(1) & ",  " & Info(0)
        Label1(4).Caption = Info(2)
    End With
End Sub
```

解　説

1

API関数「GetVersionEx」は、Windowsのバージョン番号とWindowsの種類（NT、95、98など）を取得する関数です。

引数に設定した構造体「OSVERSIONINFO」を指定して実行すると、ここにいくつかの情報を格納します。

```
Public Declare Function GetVersionEx Lib "kernel32" _
    Alias "GetVersionExA" _
    (lpVersionInformation As OSVERSIONINFO) As Long
```

構造体「OSVERSIONINFO」は、Visual Basicでは次のようなユーザー定義のデータ型として再定義して使用します。

```
Type OSVERSIONINFO
        dwOSVersionInfoSize As Long
        dwMajorVersion As Long
        dwMinorVersion As Long
        dwBuildNumber As Long
        dwPlatformId As Long
        szCSDVersion As String * 128
End Type
```

パラメータ	説明
dwOSVersionInfoSize	「OSVERSIONINFO」が使用するバイトサイズを設定します。このパラメータは、「GetVersionEx」を呼び出す前に設定しておく必要があります。
dwMajorVersion	Windowsのメジャーバージョン番号を格納します。
dwMinorVersion	Windowsのマイナー番号（メジャーバージョンの次の番号）を格納します。
dwBuildNumber	ビルドナンバーを格納します。WindowsNT4.0ではバージョン番号の次にリビジョンを表す番号として使われています。
dwPlatformId	Windowsの種類を格納します。 値　種類 0　Win32s on Windows 3.1 1　Win32 on Windows 95、Windows98 2　Windows NT
szCSDVersion	上記項目以外の、Windowsに関するインフォメーションが格納されます。この変数だけ、128バイトの固定長の文字列型が与えられています。WindowsNT4.0などでは、ServicePackをインストールしていると、ここにその情報が格納されます。

2　API関数「GetSystemInfo」は、コンピュータに組み込まれているCPUに関する情報を取得する関数です。引数に、構造体「SYSTEM_INFO」を指定して実行すると、いくつかの情報をこの構造体に格納します。

```
Public Declare Sub GetSystemInfo Lib "kernel32" _
    (lpSystemInfo As SYSTEM_INFO)
```

構造体「SYSTEM_INFO」は、Visual Basicでは次のようなユーザー定義のデータ型になります。

```
Type SYSTEM_INFO
        dwOemID As Long
        dwPageSize As Long
        lpMinimumApplicationAddress As Long
        lpMaximumApplicationAddress As Long
        dwActiveProcessorMask As Long
        dwNumberOrfProcessors As Long
        dwProcessorType As Long
        dwAllocationGranularity As Long
        dwReserved As Long
End Type
```

パラメータ	説明
dwOemID	製造会社（OEM）ごとの識別が格納されます。
dwPageSize	システムのページサイズを格納します。
lpMinimumApplicationAddress	アプリケーションとDLLの最小メモリアドレスが格納されます。
lpMaximumApplicationAddress	アプリケーションとDLLの最大メモリアドレスが格納されます。
dwActiveProcessorMask	プロセッサのコンフィギュレーションが格納されます。
dwNumberOrfProcessors	組み込まれているCPUの数が格納されます。
dwProcessorType	CPUのタイプが格納されます。Windows95/98では、「INTEL_386」「INTEL_486」「INTEL_PENTIUM」が、WindowsNT4.0では「INTEL_386」「INTEL_486」「INTEL_PENTIUM」「ALPHA_21046」が格納されます。
dwAllocationGranularity	メモリの割り当て単位が格納されます。
dwReserved	予約済み

3 このプログラムでは、これらのAPI関数を使って、WindowsとCPUの情報を取得します。プログラムの処理は、フォームに配置した［System Information］ボタンのClickイベントプロシージャで行います。

まず、ラベルコントロールに表示するメッセージを格納するための変数を2つ宣言します。変数「Info」は、同じような変数をいくつも宣言するのが面倒なので、3つの要素を持つ配列で宣言しています。

```
Private Sub Command1_Click()
    Dim Msg1 As String, Msg2 As String
    Dim Info(2) As String
```

そして、再定義したユーザー定義のデータ型を、変数に割り当てて使用できるようにしています。

```
Dim SysInfo As SYSTEM_INFO
Dim VInfo As OSVERSIONINFO
```

4 次に、「GetVersionEx」関数を実行しますが、その前にユーザー定義型変数「VInfo」の要素の1つ、「dwOSVersionInfoSize」にこの変数が使用する総バイト数を設定します。

Len関数に変数名を設定して実行すれば、その変数が消費するメモリサイズがすぐに算出できますので、これを「dwOSVersionInfoSize」に代入します。

```
VInfo.dwOSVersionInfoSize = Len(VInfo)
```

そして、「GetVersionEx」関数を実行します。

```
GetVersionEx VInfo
```

5 「GetVersionEx」関数が実行されると、ユーザー定義型変数「VInfo」の各要素に情報が格納されますので、これらを取り出してラベルコントロールに出力します。

まず、メジャーとマイナーの番号を組み合わせて、バージョン番号を作成し、最初のラベルコントロール（Label1(0)）に設定します。

```
With VInfo
    Msg1 = "バージョン " & .dwMajorVersion & "." & _
    .dwMinorVersion

Label1(0).Caption = Msg1
```

次に、「wBuildNumber」からビルドナンバーを取り出して、ラベルコントロール（Label1(1)）に設定します。

```
Label1(1).Caption = "BUILD : " & .dwBuildNumber
```

6 今度は、「dwPlatformId」の値を使って、どのWindowsなのかを表示します。Select Caseステートメントで、値を使って振り分けます。値とそれぞれのWindowsの種類は、以下のとおりです。

値	種類
0	Win32s on Windows 3.1
1	Win32 on Windows 95/Windows 98
2	Windows NT

この値を元に、文字列を作成してラベルコントロール（Label1(2)）に設定します。このとき、使用しているコンピュータのOSがWindows95/98の場合は、いずれも「dwPlatformId」には「1」が返されます。しかし、Windows95では「dwMinorVersion」に「0」が、Windows98では「1」が返されますので、この値を使ってWindows95と98を区別します。

```
Select Case .dwPlatformId
            Case 0
                Msg2 = "Win32s on Windows3.1 "
            Case 1
                If .dwMinorVersion > 0 Then
                    Msg2 = "Windows98"
                Else
                    Msg2 = "Windows95"
                End If
            Case 2
                Msg2 = "WindowsNT"
End Select
Label1(2).Caption = Msg2
```

最後に、「szCSDVersion」の値を取り出して、ラベルコントロール（Label1(3)）に設定すれば、バージョン関係の情報に対する処理は終了です。

```
Label1(3).Caption = .szCSDVersion
```

取り出したバージョン番号情報（Windows98の場合）

ワンポイントアドバイス

「szCSDVersion」は、インストールしているWindowsによっては、意味のないデータになったり、表示されない場合があります。

7 続いて、「GetSystemInfo」関数を実行します。この関数は、引数にユーザー定義型変数を設定して実行するだけです。

```
GetSystemInfo SysInfo
```

実行後に変数に格納された各値を取り出して加工します。「dwNumberOrfProcessors」は、組み込まれているCPUの数を格納していますので、これで個数を表示します。

```
With SysInfo
    Info(0) = .dwNumberOrfProcessors & " Processor"
```

また、「dwProcessorType」はCPUのタイプを格納している変数です。CPUが「Pentium」の場合は、「586」という値を返してきますから、「Pentium」という文字に置き換えます。

そして、これらを連結して、ラベルコントロールに設定します。

```
Select Case SysInfo.dwProcessorType
    Case 586
        Info(1) = "Pentium"
End Select

Info(2) = "CPU Type: " & Info(1) & ",  " & Info(0)
Label1(4).Caption = Info(2)
```

まとめ

２つの情報取得のAPI関数を使用しました。関数によっては、引数の構造体の使い方がAPIビューアの情報と異なる場合があるので、APIリファレンスをよく読んで使用するようにしてください。

Windowsのバージョン情報などは、筆者はWindows98とWindowsNT4.0の両方を使用していますので、両方の動作確認ができますが、CPUタイプなどはPentiumしか持っていないので、なかなかできません。広く配布するプログラムの場合は、可能な限り多くの機種で動作確認を行ってから、リリースするようにしたいですね。

練習問題

Q1　これまで作成した、「コンピュータ名の取得」プログラムと、「メモリの状況」プログラムを組み合わせて、総合的な情報提供プログラムを作成してください。

Form1

コンピュータ名:SETO
総物理メモリ量: 133MB
使用可能メモリ量: 40MB
総仮想メモリ量: 471MB
利用可能な仮想メモリ量: 411MB
Windows98
バージョン 4.10
BUILD : 67766408
CPU Type: Pentium, 1 Processor

完成したプログラム

解答は巻末に→

Lesson 29

第９日
２時限目

APIを使ったアプリケーションの作成

Lesson29.vbp
VB6CD2
Lesson29

壁紙チェンジャーの作成

Visual Basicが標準で装備しているコントロールやメソッド、関数などで操作できないものの１つに、Windowsのデスクトップがあります。たとえば、デスクトップの壁紙を変えるプログラムなどは、Visual Basicだけでは作成できません。そこで、Win32APIを使って、簡単に壁紙を変えられるプログラムを作成することにしました。このプログラムは、コモンダイアログボックスを使って選択したビットマップファイルを、フォームに表示するとともに、デスクトップの壁紙として設定します。

今回作成するプログラム

壁紙チェンジャー
壁紙の設定
壁紙の選択
C:¥WINDOWS¥花見.bmp
設定
壁紙の削除
終了

壁紙ファイルを選択し…

ここをクリックすると…

デスクトップに壁紙が設定される

POINT

API関数「SystemParametersInfo」は、デスクトップを構成する各部の情報を取得、設定する関数です。この関数を使うと、デスクトップの壁紙を入れ替えることができます。

壁紙用のビットマップの選択にはコモンダイアログボックスの［ファイルを開く］ダイアログボックスを使用します。

コントロールとプロパティの設定

※新しいプロジェクトファイル（標準EXE）を用意します。

1 フォームを広げ、[Caption] プロパティを「壁紙チェンジャー」に変更する

プロパティ - Form1

Form1 Form

全体 | 項目別

(ｵﾌﾞｼﾞｪｸﾄ名)	Form1
Appearance	1 - 3D
AutoRedraw	False
BackColor	&H8000000F&
BorderStyle	2 - 可変
Caption	壁紙チェンジャー
ClipControls	True
ControlBox	True
DrawMode	13 - Copy Pen
DrawStyle	0 - 実線 (中心線)
DrawWidth	1

1 ここを変更

2 ツールパレットに「Microsoft Common Dialog Control 6.0」コントロールを追加する

CommonDialog

ツールパレットに追加されたコントロール

3 フレームコントロールを2つ配置し、各コントロールの [Caption] プロパティを変更する

コントロール	[Caption] プロパティの値
Frame1	壁紙の設定
Frame2	(なし)

コンポーネントを追加する

コンポーネントを追加する方法については、「Lesson5 ツールバーコントロールの組み込み」を参照してください。

フレームコントロールを配置する

フレームコントロールを配置するには、ツールパレットの [Frame] を使います。

コマンドボタンを配置する

コマンドボタンを配置するには、ツールパレットの［CommandButton］を使います。

ラベルコントロールを配置する

ラベルコントロールを配置するには、ツールパレットの［Label］A を使います。

4 コマンドボタンをフレーム内に２つ、外に２つ配置し、プロパティとオブジェクト名を変更する

コントロール	オブジェクト名	［Caption］プロパティ
Command1	SelBtn	壁紙の選択
Command2	SetBtn	設定
Command3	DelBtn	壁紙の削除
Command4	ExitBtn	終了

配置してプロパティを変更したコマンドボタンコントロールとフレームコントロール

5 ラベルコントロールをフレーム内に配置し、次のプロパティを設定する

プロパティ	値
BorderStyle	1 ー実線
Caption	（なし）

配置してプロパティを設定したラベルコントロール

6 イメージコントロールを内側のフレームに配置し、[Strech] プロパティを「True」に設定する

7 プロジェクトに標準モジュールを追加する

8 APIビューアを起動し、データベースファイル「WIN32API.MDB」を読み込む

イメージコントロールを配置する

イメージコントロールを配置するには、ツールパレットの［Image］を使います。

標準モジュールを挿入する

［プロジェクト］メニューの［標準モジュールの追加］を選択して、標準モジュールを挿入することもできます。

［標準モジュールの追加］ダイアログボックスが表示されるので、［標準モジュール］アイコンをクリックして、［開く］をクリックします。

APIビューアを起動する

APIビューアを起動するには、タスクバーの［スタート］ボタン→［プログラム］→［Microsoft Visual Basic 6.0］→［Microsoft Visual Basic 6.0 Tools］→［APIビューア］を選択します。

APIのコードをコピーする

APIのコードを標準モジュールにコピーする詳しい方法については、「Lesson 26 APIビューアを使う」を参照してください。

9 関数と定数をコピーし、Visual Basicに戻って標準モジュールに貼り付ける

種類	有効な項目
[関数]	SystemParametersInfo
[定数]	SPI_SETDESKWALLPAPER SPIF_UPDATEINIFILE

1 ここで［関数］、［定数］を選択

2 ［有効な項目］を選択し、［追加］をクリック

3 ここをクリック

標準モジュールに貼り付けられたコード

10 ツールパレットの［CommonDialog］を使って、コモンダイアログコントロールを配置する

1 斜めにドラッグして配置

11 ［プロパティページ］の［開く/保存］タブで、次の項目を設定し、［OK］ボタンをクリックする

項目	値
タイトル	壁紙の選択
初期ディレクトリ	c:¥
フィルタ	ビットマップファイル (*.*)¦ *.bmp

12 ［設定］ボタンの［Enabled］プロパティを「False」に設定する

［プロパティページ］ダイアログ

［プロパティページ］ダイアログボックスを表示するには、コントロールを選択し、プロパティウィンドウの［(プロパティページ)］をクリックし、右欄の［...］ボタンをクリックします。

コードの記述

Module1:(General)-(Declarations)

Lesson29 - Module1 (コード)

(General) / (Declarations)

```
Public Declare Function SystemParametersInfo Lib "user32" _
    Alias "SystemParametersInfoA" (ByVal uAction As Long, _
    ByVal uParam As Long, ByVal lpvParam As Any, _
    ByVal fuWinIni As Long) As Long

Public Const SPI_SETDESKWALLPAPER = 20
Public Const SPIF_UPDATEINIFILE = &H1
```

Form1:(General)-(Declarations)

Lesson29 - Form1 (コード)

(General) / (Declarations)

```
Dim Fname As String
```

Form1:SelBtn-Click

Lesson29 - Form1 (コード)

SelBtn / Click

```
Private Sub SelBtn_Click()
    With CommonDialog1
        .ShowOpen
        Fname = .filename
    End With

    If Fname = "" Then
        Exit Sub
    End If
    Image1.Picture = LoadPicture(Fname)
    SetBtn.Enabled = True
    Label1.Caption = Fname
End Sub
```

Form1:SetBtn-Click

Lesson29 - Form1 (ｺｰﾄﾞ)

SetBtn | Click

```
Private Sub SetBtn_Click()
    Dim Ret As Long

    Ret = SystemParametersInfo(SPI_SETDESKWALLPAPER, 0, _
        Fname, SPIF_UPDATEINIFILE)

    If Ret = False Then
        MsgBox "壁紙の設定に失敗しました。"
    End If
End Sub
```

Form1:ExitBtn-Click

Lesson29 - Form1 (ｺｰﾄﾞ)

ExitBtn | Click

```
Private Sub ExitBtn_Click()
    Unload Me
    End
End Sub
```

Form1:DelBtn-Click

Lesson29 - Form1 (ｺｰﾄﾞ)

DelBtn | Click

```
Private Sub DelBtn_Click()
    Dim Ret As Long

    Ret = SystemParametersInfo(SPI_SETDESKWALLPAPER, 0, _
        "", SPIF_UPDATEINIFILE)

    If Ret = False Then
        MsgBox "壁紙の設定に失敗しました。"
    End If
End Sub
```

ワンポイントアドバイス

APIビューワを使用して、API関数「SystemParametersInfo」の定義をそのまま標準モジュールにコピーすると、プログラムは正常に動作しません。これは、関数の定義の記述に間違いがあるためです。

API関数「SystemParametersInfo」の引数のひとつ「ByRef lpvParam As Any」の記述を、「ByVal lpvParam As Any」に修正してください。

【修正前】

```
Public Declare Function SystemParametersInfo Lib "user32" _
    Alias "SystemParametersInfoA" (ByVal uAction As Long, _
    ByVal uParam As Long, _
    ByRef lpvParam As Any, _ ←この部分が記述間違い
    ByVal fuWinIni As Long) As Long
```

【修正後】

```
Public Declare Function SystemParametersInfo Lib "user32" _
    Alias "SystemParametersInfoA" (ByVal uAction As Long, _
    ByVal uParam As Long, _
    ByVal lpvParam As Any, _ ←ByRefをByValに修正する
    ByVal fuWinIni As Long) As Long
```

解説

1 API関数「SystemParametersInfo」は、デスクトップに関する大変多くの項目を設定することができる関数です（巻末の付録「SystemParametersInfoについて」を参照）。このAPI関数は、4つの引数を持ちます。

主に「uAction」に、操作したい内容を定数で設定し、実行します。たとえば、定数「SPI_GETBEEP」を設定して関数を実行すると、コンピュータのビープ音の設定がオンになっていれば、「lpvParam」に「True」を返してきます。

一方、定数「SPI_SETBEEP」を「uAction」に設定し、「uParam」に「False」を設定して関数を実行すると、コンピュータのビープ音をオフにします。

このように、「Get」が付く定数は設定値の取得を、「Set」が付く定数は設定を行う定数になっています。これらの定数を、引数「uAction」に設定して関数を実行することで、デスクトップの情報の取得･設定を行います。

```
Declare Function SystemParametersInfo Lib "user32" _
    Alias "SystemParametersInfoA" (ByVal uAction As Long, _
    ByVal uParam As Long, ByVal lpvParam As Any, _
    ByVal fuWinIni As Long) As Long
```

「uAction」の定数値「SPI_SETDESKWALLPAPER」は、もう1つの引数「lpvParam」に、ビットマップファイルのファイル名をフルパスで設定すると、そのファイルを壁紙に設定します。そして、最後の引数「fuWinIni」に、定数「SPIF_UPDATEINIFILE」を設定すると、Windowsの壁紙設定をレジストリに保存します（引数「uParam」は使用しませんので、「0」を設定します）。この2つの定数を使用する場合は、これらも標準モジュールで再定義しておく必要があります。

```
Public Const SPI_SETDESKWALLPAPER = 20
Public Const SPIF_UPDATEINIFILE = &H1
```

2　このプログラムは、コモンダイアログコントロールの［ファイルを開く］ダイアログボックスと組み合わせて、選択したビットマップファイルを壁紙に設定します。

①［壁紙の選択］ボタンをクリックすると、［ファイルを開く］ダイアログボックスが表示されます。ダイアログボックスは、拡張子が「bmp」のファイルだけ表示されるように設定してあります。

②ユーザーがビットマップファイルを選びダイアログボックスを閉じると、ラベルコントロールにそのファイル名をフルパスで表示します。それと同時に、イメージコントロールにその画像を表示します。しかし、この時点ではまだ壁紙は変更しません。

③［設定］ボタンをクリックすると、選んだファイルを壁紙に設定します。ここではじめて、API関数「SystemParametersInfo」が実行されます。

選択したファイルのフルパスと画像が表示される

ここをクリックすると…

壁紙が設定される

④「壁紙の削除」ボタンをクリックすると、ファイル名を指定せずに（空白のまま）API関数「SystemParametersInfo」を実行します。ファイル名を指定していませんので、壁紙はデスクトップから消去されます。

3 では、コードを作成しましょう。最初に、ビットマップファイルのファイル名を格納する変数「Fname」を、モジュールレベルの変数で宣言しておきます。

```
Dim Fname As String
```

次に、［壁紙の選択］ボタンがクリックされたときの処理を作成します。このボタンは、ビットマップファイルを選択する処理を行います。まず、コモンダイアログコントロールの「ShowOpen」メソッドを実行して、［ファイルを開く］ダイアログボックスを表示します。そして、［filename］プロパティから、ユーザーが選択したファイル名を取得し、変数「Fname」に格納しておきます。

```
Private Sub SelBtn_Click()
    With CommonDialog1
        .ShowOpen
        Fname = .filename
    End With
```

そして、もしファイル名が空白であれば、このプロシージャをここで終了します。

```
If Fname = "" Then
    Exit Sub
End If
```

4　ファイル名が取得できたら、そのファイル名をフレーム内に配置したイメージコントロールの［Picture］プロパティに、LoadPicture関数で設定します。これで、選択したビットマップファイルが表示されます。また。この時点で、［設定］ボタンを有効にし、いつでも壁紙に設定できるようにします。あわせて、ラベルコントロールの［Caption］プロパティにファイル名を設定し、選んだファイルがわかるようにしておきます。

```
Image1.Picture = LoadPicture(Fname)
SetBtn.Enabled = True
Label1.Caption = Fname
```

5　デスクトップの壁紙に設定できるイメージファイルは、ビットマップファイルだけです。

設定するビットマップファイルを取得したら、今度は［設定］ボタンでこれを壁紙に設定します。これは、2つの定数と取得したファイル名を引数に設定して、API関数「SystemParametersInfo」を実行するだけです。

これで、設定したファイル名のビットマップファイルが、壁紙に設定され、Windowsの設定として保存されます。このとき、「SystemParametersInfo」は、設定に成功すると「True」を、失敗すると「False」を返してきますので、これを変数に受け取って失敗した時はメッセージボックスを表示するようにします。

```
Private Sub SetBtn_Click()
    Dim Ret As Long

    Ret = SystemParametersInfo(SPI_SETDESKWALLPAPER, 0, _
        Fname, SPIF_UPDATEINIFILE)

    If Ret = False Then
        MsgBox "壁紙の設定に失敗しました。"
    End If
End Sub
```

6　［壁紙の削除］ボタンは、［設定］ボタンと同様に、「SystemParametersInfo」関数を実行するだけですが、ファイル名の引数に""（空白）を設定します。これで、デスクトップには何の壁紙も設定されず、壁紙を削除したことと同じことになります。

```
Private Sub DelBtn_Click()
    Dim Ret As Long

    Ret = SystemParametersInfo(SPI_SETDESKWALLPAPER, 0, _
        "", SPIF_UPDATEINIFILE)

    If Ret = False Then
        MsgBox "壁紙の設定に失敗しました。"
    End If

End Sub
```

7　最後に、［終了］ボタンの処理を作成して完成です。

```
Private Sub ExitBtn_Click()
    Unload Me
    End
End Sub
```

まとめ

API関数「SystemParametersInfo」は、壁紙だけでなくスクリーンセーバーの設定やアイコンの間隔、マウスのクリック速度など、コントロールパネルアプレットに組み込まれている機能の操作ができる関数です。それぞれ、定数を設定することでアクセスする項目を切り替えますので、使用目的に合った定数を標準モジュールで再定義してから使用してください。

練習問題

Q1 作成したプログラムを、壁紙の選択をリストボックスから行えるように修正してください。

解答は巻末に→

ワンポイントアドバイス

インターネットエクスプローラ Ver4.0 以降がインストールされた Windows では、「アクティブデスクトップ（Active Desktop）」を有効にしていると、このプログラムを使った壁紙の切り替えができません。必ず、アクティブデスクトップをオフにしてから、プログラムを使用してください。

画面のプロパティ

背景 | スクリーン セーバー | デザイン | 効果 | Web | 設定

Active Desktop を Web ページとして表示(V)
Internet Explorer チャンネル バー
新規(N)... 削除(D) プロパティ(P) すべてリセット(R)
デスクトップ アイコンのクリック方法を変更します: フォルダ オプション(F)
OK キャンセル 適用(A)

Lesson 30

第９日
３時限目

APIを使ったアプリケーションの作成

Lesson30.vbp
VB6CD2
Lesson30

自動壁紙チェンジャーの作成

このレッスンでは、壁紙を変更するプログラムの応用プログラムを作成します。使用するAPI関数は、前のレッスンと同じものを使用しますが、今度はタイマーコントロールを使って、登録したビットマップファイルを順番に選び、一定時間ごとに壁紙を入れ替えるプログラムです。テクニック的にはさほど難しくなく、今日まで学んできたテクニックを組み合わせれば、簡単にできあがるプログラムです。

今回作成するプログラム

複数のビットマップファイルを指定しておくと...

一定時間ごとに壁紙が変更される

POINT

壁紙に設定するビットマップファイル名は、リストボックスコントロールに組み込んでおきます。そして、タイマーコントロールの「Timer」イベントで、リストボックスから順番にファイル名を取得し、API関数「SystemParametersInfo」を実行します。

すべての処理は、タイマーコントロールの「Timer」イベントで行うのがポイントです。

コントロールとプロパティの設定

※新しいプロジェクトファイル（標準EXE）を用意します。

1 フォームを広げ、[Caption] プロパティを「自動壁紙チェンジャー」に変更する

プロパティ - Form1

Form1 Form

全体 | 項目別

(オブジェクト名)	Form1
Appearance	1 - 3D
AutoRedraw	False
BackColor	&H8000000F&
BorderStyle	2 - 可変
Caption	自動壁紙チェンジャー
ClipControls	True
ControlBox	True
DrawMode	13 - Copy Pen
DrawStyle	0 - 実線 (中心線)
DrawWidth	1
Enabled	True

1 ここを変更

2 ラベルコントロールを2つ配置し、各コントロールの [Caption] プロパティを変更する

ラベルコントロール	[Caption] プロパティ
Label1	登録ビットマップ
Label2	現在実行中の壁紙

自動壁紙チェンジャー

登録ビットマップ

現在実行中の壁紙

配置してプロパティを設定したラベルコントロール

ラベルコントロールを配置する

ラベルコントロールを配置するには、ツールパレットの［Label］A を使います。

リスト項目の設定

リストボックスのリスト項目を設定するとき、プロパティウィンドウの［List］プロパティに、自分のコンピュータのハードディスクにあるビットマップファイルをフルパスで記述してください。

ここでは例としてデフォルトで「Windows」フォルダに存在しているはずのビットマップを設定しています。

存在しないファイルを指定すると、後で設定する「SystemParametersInfo」関数が実行されたときに、壁紙に何も設定されません。フルパスは正しく設定してください。

3 ツールパレットの［ListBox］を使ってリストボックスを配置し、［List］プロパティを設定する

自動壁紙チェンジャー

登録ビットマップ

c:¥windows¥エジプト.bmp
c:¥windows¥しゃぼん玉.bmp
c:¥windows¥トライアングル.bmp
c:¥windows¥バンブー.bmp

現在実行中の壁紙

配置してプロパティを設定したリストボックスコントロール

ワンポイントアドバイス

本書で使用している、Windowsに付属するビットマップファイルをリストに組み込む場合は、カタカナの半角・全角文字に注意してください。

Windows95とWindows98では、これらのビットマップファイルは同じファイル名に見えますが、実際は半角記述と全角記述の違いがあります。

Windows95でのファイル名	Windows98でのファイル名
ｴｼﾞﾌﾟﾄ.bmp	エジプト.bmp
ﾄﾗｲｱﾝｸﾞﾙ.bmp	トライアングル.bmp
ﾊﾞﾝﾌﾞｰ.bmp	バンブー.bmp

4 ツールパレットの［Label］を使ってラベルコントロールを配置し、次のプロパティを設定する

プロパティ	プロパティ値
BorderStyle	1 －実線
Caption	（なし）

自動壁紙チェンジャー

登録ビットマップ

c:¥windows¥エジプト.bmp
c:¥windows¥しゃぼん玉.bmp
c:¥windows¥トライアングル.bmp
c:¥windows¥バンブー.bmp

現在実行中の壁紙

配置してプロパティを設定したラベルコントロール

5 コマンドボタンを配置し、オブジェクト名と[Caption] プロパティを変更する

コントロール	オブジェクト名	[Caption] プロパティ
Command1	ExitBtn	終了

現在実行中の壁紙

終了

配置してプロパティを設定したコマンドボタンコントロール

6 タイマーコントロールを配置し、[Interval] プロパティを「5000」に設定する

自動壁紙チェンジャー

登録ビットマップ

c:¥windows¥エジプト.bmp
c:¥windows¥しゃぼん玉.bmp
c:¥windows¥トライアングル.bmp
c:¥windows¥バンブー.bmp

現在実行中の壁紙

終了

配置してプロパティを設定したタイマーコントロール

7 プロジェクトに標準モジュールを追加する

Project1 - Microsoft Visual Basic [デザイン]

ファイル(F) 編集(E) 表示(V) プロジェクト(P) 書式(O) デバッグ(D)

フォーム モジュール(F)
MDI フォーム モジュール(I)
標準モジュール(M)
クラス モジュール(C)
ユーザー コントロール(U)
プロパティ ページ(R)

1 ここをクリック

2 ここをクリック

コマンドボタンを配置する

コマンドボタンを配置するには、ツールパレットの［CommandButton］を使います。

タイマーコントロールを配置する

タイマーコントロールを配置するには、ツールパレットの［Timer］を使います。

標準モジュールを挿入する

［プロジェクト］メニューの［標準モジュールの追加］を選択して、標準モジュールを挿入することもできます。

［標準モジュールの追加］ダイアログボックスが表示されるので、［標準モジュール］アイコンをクリックして、［開く］をクリックします。

APIビューアを起動する

APIビューアを起動するには、タスクバーの［スタート］ボタン→［プログラム］→［Microsoft Visual Basic 6.0］→［Microsoft Visual Basic 6.0 Tools］→［APIビューア］を選択します。

APIのコードをコピーする

APIのコードを標準モジュールにコピーする詳しい方法については、「Lesson 26 APIビューアを使う」を参照してください。

8 APIビューアを起動し、次の関数と定数を追加し、Visual Basicに戻って標準モジュールに貼り付ける

種類	有効な項目
［関数］	SystemParametersInfo
［定数］	SPI_SETDESKWALLPAPER SPIF_UPDATEINIFILE

1 ここで［関数］、［定数］を選択

2 ［有効な項目］を選択し、［追加］をクリック

3 ここをクリック

標準モジュールに貼り付けられたコード

```
Public Declare Function SystemParametersInfo Lib "user32" A
Public Const SPI_SETDESKWALLPAPER = 20
Public Const SPIF_UPDATEINIFILE = &H1
```

ワンポイントアドバイス

APIビューワでAPI関数「SystemParametersInfo」の定義を標準モジュールにコピーしたら、引数のひとつ「ByRef lpvParam As Any」の記述を、「ByVal lpvParam As Any」に修正してください。

```
Public Declare Function SystemParametersInfo Lib "user32" _
    Alias "SystemParametersInfoA" (ByVal uAction As Long, _
    ByVal uParam As Long, _
    ByVal lpvParam As Any, _  ←ByRefをByValに修正する
    ByVal fuWinIni As Long) As Long
```

コードの記述

Module1:(General)-(Declarations)

Project1 - Module1 (コード)

(General) / (Declarations)

```
Declare Function SystemParametersInfo Lib "user32" _
    Alias "SystemParametersInfoA" (ByVal uAction As Long, _
    ByVal uParam As Long, ByVal lpvParam As Any, _
    ByVal fuWinIni As Long) As Long

Public Const SPI_SETDESKWALLPAPER = 20
Public Const SPIF_UPDATEINIFILE = &H1
```

Form1:(General)-(Declarations)

Project1 - Form1 (コード)

(General) / (Declarations)

```
Dim i As Integer
```

Form1:Form-Load

Project1 - Form1 (コード)

Form / Load

```
Private Sub Form_Load()
    i = 0
End Sub
```

Form1:Timer1-Timer

Project1 - Form1 (コード)

Timer1 / Timer

```
Private Sub Timer1_Timer()
    Dim Ret As Long
    Dim Fname As String

    Fname = List1.List(i)
    List1.Selected(i) = True
    Ret = SystemParametersInfo(SPI_SETDESKWALLPAPER, 0, _
        Fname, SPIF_UPDATEINIFILE)
    If Ret = False Then
        MsgBox "壁紙の設定に失敗しました。"
        Exit Sub
    End If

    i = i + 1
    If i = 4 Then
```

```
        i = 0
    End If
    Label2.Caption = Fname
End Sub
```

Form1:ExitBtn-Click

Project1 - Form1 (コード)

ExitBtn | Click

```
Private Sub ExitBtn_Click()
    Unload Me
    End
End Sub
```

解説

1 API関数「SystemParametersInfo」については、すでに前のレッスンで説明しましたので、プログラムの動作について説明します。

①まず、壁紙に設定するビットマップファイルを、リストボックスコントロールの［List］プロパティに設定します。

②次に、タイマーコントロールの「Timer」イベントが発生すると、このリスト項目を1つ読み込んで、「SystemParametersInfo」関数に渡します。そして、この関数を実行します。

ビットマップファイルが順番に壁紙に設定される

自動壁紙チェンジャー

登録ビットマップ

c:¥windows¥エジプト.bmp
c:¥windows¥しゃぼん玉.bmp
c:¥windows¥トライアングル.bmp
c:¥windows¥バンブー.bmp

現在実行中の壁紙

c:¥windows¥トライアングル.bmp

終了

③それと同時に、リストボックスコントロールの、同じファイル名の項目を選択状態にし、ラベルコントロールにも出力して、現在壁紙に設定されているファイル名がわかるようにします。

④タイマーコントロールは、［Interval］プロパティを「5000」ミリ秒（5秒）に設定してありますので、5秒ごとにリストボックスのファイル名が読み込まれ、壁紙が変更されます。

2　では、コードを作成していきましょう。最初に、フォームのコードウィンドウに、モジュールレベルの変数iを宣言します。この変数は、リストボックスのリスト項目を参照するインデックス番号として使用します。

```
Dim i As Integer
```

宣言したら、フォームのLoadイベントプロシージャで「0」に初期化します。これはリストボックスのリスト項目のインデックスが「0」から始まるからで、「Timer1_Timer()」プロシージャに処理が移ったときに、リスト項目の1番目から処理を行うための準備です。プログラム起動時の処理はこれだけです。

```
Private Sub Form_Load()
    i = 0
End Sub
```

3　今度は、タイマーコントロールのTimerイベントプロシージャの処理を作成します。このプログラムのメインの処理は、すべてこのプロシージャで行います。

まず、変数を2つ宣言します。1つは、API関数「SystemParametersInfo」の戻り値を格納する変数「Ret」で、もう1つはファイル名を格納する変数「Fname」です。

```
Private Sub Timer1_Timer()
    Dim Ret As Long
    Dim Fname As String
```

次に、リストボックスの先頭のリスト項目を、変数「Fname」に格納します。これで、壁紙にするファイル名が取得できました。そして、その項目を［Selected］プロパティを使用して選択状態にします。

```
Fname = List1.List(i)
List1.Selected(i) = True
```

自動壁紙チェンジャー

登録ビットマップ

c:¥windows¥エジプト.bmp
c:¥windows¥しゃぼん玉.bmp
c:¥windows¥トライアングル.bmp
c:¥windows¥バンブー.bmp

現在実行中の壁紙

c:¥windows¥トライアングル.bmp

ファイル名を取得し、1番目の項目を選択状態にしておく

4 壁紙にするファイル名が決まったら、「SystemParametersInfo」関数を実行します。関数の戻り値を受け取り、エラー処理を組み立てます。

```
Ret = SystemParametersInfo(SPI_SETDESKWALLPAPER, 0, _
    Fname, SPIF_UPDATEINIFILE)

If Ret = False Then
    MsgBox "壁紙の設定に失敗しました。"
    Exit Sub
End If
```

そして、リスト項目のインデックス用変数「i」を1つ増やします。これで、次にTimerイベントが発生すれば、2番目のリスト項目が読み込まれ、壁紙が変更されます。

変数「i」の値が「4」になったら、「i」を「0」に初期化していますが、これは変数「i」の値が、設定したリスト項目以上の値にならないようにするための処置です。この例題プログラムでは、リストボックスに4つの項目しか設定していないため、「If i = 4 Then」という式にしていますが、項目数が多ければ、この数字もそれに合わせて増やしてください。

```
i = i + 1
If i = 4 Then
    i = 0
End If
```

この処理で、リストボックスに設定したファイル名が、順番に取り出されて、「SystemParametersInfo」関数が実行されます。

壁紙の設定が成功したら、ファイル名をラベルコントロールに出力します。

```
Label2.Caption = Fname
```

5　最後に、［終了］ボタンに、プログラムの終了処理を作成して完成です。

```
Private Sub ExitBtn_Click()
    Unload Me
    End
End Sub
```

まとめ

タイマーコントロールの「Timer」イベントを使うと、このような自動壁紙チェンジャーが作成できます。壁紙に使うビットマップファイル名の設定に、リストボックスコントロールを使用しましたが、配列などの変数を使ってもかまいません。なお、タイマーコントロールの［Interval］プロパティは、最大65,535ミリ秒までしか設定できませんので、注意してください。

練習問題

Q1　作成したプログラムを、指定した時刻になると壁紙が切り替わるような機能に修正してください。例えば、15時になると切り替わるようにしてください。

【ヒント】
システムから時刻を取得するには、「Time」関数を使用します。

解答は巻末に→

ワンポイントアドバイス

プログラムの起動時に、タイマーのインターバルを待たずに、1番目のリスト項目を壁紙に設定したいなら「Private Sub Form_Load()」プロシージャにも、以下のコードを追加すると良いでしょう。

```
List1.Selected(i) = True
Ret = SystemParametersInfo(SPI_SETDESKWALLPAPER, 0, _
    List1.List(i), SPIF_UPDATEINIFILE)
If Ret = False Then
    MsgBox "壁紙の設定に失敗しました。"
    Exit Sub
End If
```

Lesson 31

第９日
４時限目

APIを使ったアプリケーションの作成

Lesson31.vbp
VB6CD2
Lesson31

ディスクの空き容量を把握する

今度は、Windowsのファイルシステムを操作するAPI関数を使用して、マイコンピュータのような、ディスクの使用量や空き容量を表示するプログラムを作成します。API関数「GetDiskFreeSpace」を使用すると、ディスクの容量に関する情報を取得できます。この関数を、Visual Basicのドライブリストボックスコントロールと組み合わせて、指定したドライブの情報を表示するようにします。また、Visual Basicのチャートコントロールを使用して、ディスクの使用量と空き容量を円グラフで表示する機能も組み込みます。

今回作成するプログラム

Drive Space Checker
i: [WINDOWS]
Check
Space size (Byte)
Total 1077313536
Used 928612352
Free 148701184
Sector Size 512
DiskSpace
終了

ここをクリックすると…

ディスクの使用量と空き容量が円グラフで表示される

POINT

API関数「GetDiskFreeSpace」は、指定したドライブの空き容量や総容量を、クラスタとセクタのサイズで返してくる関数です。このデータを元に、式でバイト数に変換し、フォームに配置したラベルコントロールで表示します。また、チャートコントロールを使って、データをグラフ表示するテクニックも使用します。

コントロールとプロパティの設定

※新しいプロジェクトファイル（標準EXE）を用意します。

1 ツールパレットに「Microsoft Chart Control」コントロールを追加する

ツールパレットに追加されたコントロール

2 フォームを広げ、［Caption］プロパティを「Drive Space Checker」に変更する

プロパティ - Form1

Form1 Form

全体 | 項目別

(オブジェクト名)	Form1
Appearance	1 - 3D
AutoRedraw	False
BackColor	&H8000000F&
BorderStyle	2 - 可変
Caption	Drive Space Checker
ClipControls	True
ControlBox	True

1 ここを変更

3 ツールパレットの［DriveListBox］ を使って、ドライブリストボックスコントロールを配置する

Drive Space Checker

1 斜めにドラッグして配置

コンポーネントを追加する

コンポーネントを追加する方法については、「Lesson5 ツールバーコントロールの組み込み」を参照してください。

フレームコントロールを配置する

フレームコントロールを配置するには、ツールパレットの［Frame］を使います。

ラベルコントロールを配置する

ラベルコントロールを配置するには、ツールパレットの［Label］Aを使います。

4 フレームコントロールを配置し、［Caption］プロパティを「Space size (Byte)」に変更する

5 フレーム内にラベルコントロールを４つ配置し、［Caption］プロパティを変更する

フレームコントロール	［Caption］プロパティ
Label1	Total
Label2	Used
Label3	Free
Label4	Sector Size

6 再度［Label］ を使って、枠線付きのラベルコントロールを４つ配置する

プロパティ	プロパティの値
Alignment	３－右揃え
BorderStyle	１－実線
Caption	（なし）

Drive Space Checker
c:
Space size (Byte)
Total
Used
Free
Sector Size

配置してプロパティを設定したラベルコントロール

7 ツールパレットの［MSChart］ を使ってチャートコントロールを配置し、プロパティを設定する

プロパティ	値
ColumnCount	2
ChartType	14 － VtChChartType2dPie
Data	100
RowCount	1
RowLabel	Disk Space

Drive Space Checker
c:
Space size (Byte)
Total
Used
Free
Sector Size
DiskSpace

配置してプロパティを設定したチャートコントロール

コントロールをコピーする

プロパティの設定が同じコントロールを複数作成する場合は、コントロールを1つ作成したら、それをコピーして貼り付けると効率的に作業できます。

コマンドボタンを配置する

コマンドボタンを配置するには、ツールパレットの［CommandButton］を使います。

標準モジュールを挿入する

［プロジェクト］メニューの［標準モジュールの追加］を選択して、標準モジュールを挿入することもできます。

［標準モジュールの追加］ダイアログボックスが表示されるので、［標準モジュール］アイコンをクリックして、［開く］をクリックします。

APIビューアを起動する

APIビューアを起動するには、タスクバーの［スタート］ボタン→［プログラム］→［Microsoft Visual Basic 6.0］→［Microsoft Visual Basic 6.0 Tools］→［APIビューア］を選択します。

APIのコードをコピーする

APIのコードを標準モジュールにコピーする詳しい方法については、「Lesson 26 APIビューアを使う」を参照してください。

8 コマンドボタンコントロールを2つ配置し、オブジェクト名と［Caption］プロパティを変更する

コントロール	オブジェクト名	［Caption］プロパティ
Command1	CheckBtn	Check
Command2	ExitBtn	終了

9 標準モジュールをプロジェクトに追加する

10 APIビューアを起動し、「GetDiskFreeSpace」関数を選び、標準モジュールにコピーする

11 完成したフォーム

Drive Space Checker

c:　Check

Space size (Byte)

Total

Used

Free

Sector Size

DiskSpace

終了

コードの記述

Module1:(General)-(Declarations)

Lesson31 - Module1 (ｺｰﾄﾞ)

(General)　(Declarations)

```
Declare Function GetDiskFreeSpace Lib "kernel32" _
    Alias "GetDiskFreeSpaceA" (ByVal lpRootPathName As String, _
    lpSectorsPerCluster As Long, lpBytesPerSector As Long, _
    lpNumberOfFreeClusters As Long, _
    lpTtoalNumberOfClusters As Long) As Long
```

Form1:CheckBtn-Click

Lesson31 - Form1 (ｺｰﾄﾞ)

CheckBtn　Click

```
Private Sub CheckBtn_Click()
    Dim lpRootPathName As String
    Dim lpSectorsPerCluster As Long
    Dim lpBytesPerSector As Long
    Dim lpNumberOfFreeClusters As Long
    Dim lpTtoalNumberOfClusters As Long
    Dim DriveName As String
    Dim FreeTotal As Long, TotalCapa As Long, UsedTotal As Long
```

```
    lpRootPathName = Left(Drive1.Drive, 2) & "¥"

    GetDiskFreeSpace lpRootPathName, lpSectorsPerCluster, _
        lpBytesPerSector, lpNumberOfFreeClusters, _
        lpTtoalNumberOfClusters
    TotalCapa = lpTtoalNumberOfClusters * lpSectorsPerCluster * _
        lpBytesPerSector
    FreeTotal = lpNumberOfFreeClusters * lpSectorsPerCluster * _
        lpBytesPerSector
    UsedTotal = TotalCapa - FreeTotal

    Label2(0).Caption = TotalCapa
    Label2(1).Caption = UsedTotal
    Label2(2).Caption = FreeTotal
    Label2(3).Caption = lpBytesPerSector

    With MSChart1
        .Column = 1
        .Data = CInt(FreeTotal / 1000000)
        .Column = 2
        .Data = CInt(UsedTotal / 1000000)
    End With
End Sub
```

Form1:ExitBtn-Click

Lesson31 - Form1 (コード)

ExitBtn | Click

```
Private Sub ExitBtn_Click()
    Unload Me
    End
End Sub
```

解説

1 API関数「GetDiskFreeSpace」は、指定したドライブの容量に関する情報を、システムから取得する関数です。全部で5つの引数を持ち、引数「lpRootPathName」にドライブ名を設定して関数を実行します。関数が実行されると、残りの4つの引数に情報が格納されます。

```
Declare Function GetDiskFreeSpace Lib "kernel32" _
    Alias "GetDiskFreeSpaceA" (ByVal lpRootPathName As _
    String, lpSectorsPerCluster As Long, lpBytesPerSector _
    As Long, lpNumberOfFreeClusters As Long, _
    lpTtoalNumberOfClusters As Long) As Long
```

引数	説明
lpRootPathName	情報を取得するドライブ名を指定します。
lpSectorsPerCluster	1クラスタあたりのセクタ数を格納します。
lpBytesPerSector	1セクタあたりのバイト数を格納します。
lpNumberOfFreeClusters	空きクラスタ数を格納します。
lpTtoalNumberOfClusters As Long	ディスクの総クラスタ数を格納します。

ワンポイントアドバイス

この関数は、HDDの1つのパーティションが2GBを超える場合は、正常なディスク容量を表示できません。Windows95OSR2およびWindows98に搭載された新しいファイルフォーマット「FAT32」で設定された、2GBを超えるラージパーティションのHDDの正しいディスク容量を把握するには、Windows98の新しいAPI関数「GetDiskFreeSpaceEx」を使用します。ただし、Visual Basic6.0に付属するAPIビューワではこの関数の定義が収録されていませんので、使用する場合は自分で調べる必要があります。

2　コンピュータのドライブの最小単位を「セクタ」といいます。ディスクのフォーマットの種類によって、1セクタのバイト数が決まっています。そして、「セクタ」がいくつか集まると、「クラスタ」という単位になります。したがって、ディスクの総バイト数は、次のような計算式で算出されます。

総クラスタ数 × 1クラスタあたりのセクタ数 × 1セクタあたりのバイト数

API関数「GetDiskFreeSpace」は、ディスクの容量や使用状況を、バイト数で返すのではなく、セクタ数やクラスタ数で返してきます。ですから、ディスク容量の使用状況を知るには、このように式を組み立てて計算します。たとえば、1.44MBのフロッピーディスクは、次のような数値になります。

1セクタ　　＝　　512バイト
1クラスタ　＝　　1セクタ
総クラスタ数＝　　2847

これをすべて掛け算すれば、ディスクの総バイト数になります。

2847 × 1 × 512 = 1457664

この「総クラスタ数」「1クラスタあたりのセクタ数」「1セクタのバイト数」は、ディスクの形式やフォーマットの方法、ハードディスクではパーテションの切り方などで変わってきます。

3 このディスク容量の仕組みを知った上で、実際にディスクの空き容量や使用量などを把握するコードを作成していきましょう。このプログラムでは、フォームに配置した［Check］ボタンがクリックされたときに、処理を行うようにします。まず、API関数「GetDiskFreeSpace」の引数に設定する変数を宣言します。

```
Private Sub CheckBtn_Click()
    Dim lpRootPathName As String
    Dim lpSectorsPerCluster As Long
    Dim lpBytesPerSector As Long
    Dim lpNumberOfFreeClusters As Long
    Dim lpTtoalNumberOfClusters As Long
```

変数の意味がわかりやすいように、Declareステートメントによる宣言で使用した引数と同じ名前の変数で宣言しています。

更に、ディスク容量を計算で出した合計値を、格納する変数を3つ宣言します。

```
Dim FreeTotal As Long, TotalCapa As Long, UsedTotal As Long
```

4 調べるディスクのドライブ名は、ドライブリストボックスから取得します。これは、ドライブリストボックスの［Drive］プロパティの値を調べます。ただし、このプロパティには、ドライブ名とそのボリュームラベルが以下のような形で格納されています。

ドライブ名:ボリュームラベル（例：c:MyHD）

そのため、ドライブ名だけを取り出すために、Left関数を使用します。そして、取り出したドライブ名を変数「lpRootPathName」に代入し、末尾に「¥」記号を追加します。

```
lpRootPathName = Left(Drive1.Drive, 2) & "¥"
```

5 変数「lpRootPathName」にドライブ名を設定したら、API関数「GetDiskFreeSpace」を実行します。引数には、それぞれの変数を記述します。

ワンポイントアドバイス

「¥」記号を追加しないでいると、パーティションを切ったハードディスクなどではプログラムがうまく動作しないことがあります。注意しましょう。

```
GetDiskFreeSpace lpRootPathName, _
    lpSectorsPerCluster, lpBytesPerSector, _
    lpNumberOfFreeClusters, lpTtoalNumberOfClusters
```

関数が実行されると、「lpRootPathName」以外の変数に、それぞれの情報が格納されますので、式を使ってバイト単位の容量に換算します。まず、ドライブの総バイト数を次の計算式で算出します。計算結果は、変数「TotalCapa」に代入します。

総クラスタ数×1クラスタあたりのセクタ数×1セクタあたりのバイト数

```
TotalCapa = lpTtoalNumberOfClusters * _
    lpSectorsPerCluster * lpBytesPerSector
```

また、空き容量も、同じような式で算出し、結果を変数「FreeTotal」に代入します。

```
FreeTotal = lpNumberOfFreeClusters * _
    lpSectorsPerCluster * lpBytesPerSector
```

6

総使用量は、総バイト数から空き容量を差し引けば算出できますので、答えを変数「UsedTotal」に格納します。

```
UsedTotal = TotalCapa - FreeTotal
```

計算ができたら、これらをフォームに配置したラベルコントロールに出力します。

```
Label2(0).Caption = TotalCapa
Label2(1).Caption = UsedTotal
Label2(2).Caption = FreeTotal
Label2(3).Caption = lpBytesPerSector
```

7

今度は、計算結果をチャートコントロールで円グラフにして表示します。チャートコントロールは、内部に行列をもつ［Data］プロパティに、グラフにするデータを設定すれば、そのデータを元に自動的にグラフを作成してくれるコントロールです。

データをグラフ化する場合は、最初に「ColumnCount」と「RowCount」プロパティで、データの個数を設定します。通常の2Dグラフであれば、1行多列のデータになりますから、一次元配列の設定を、「ColumnCount」と「RowCount」プロパティに設定します。たとえば、5つのデータをグラフにしたい場合は、1行5列のデータになりますから、

```
ColumnCount = 5
RowCount    = 1
```

という設定をします。これで、［Data］プロパティに一次元で5つの要素を持った配列が設定されます。

ここの配列にアクセスするには、［Column］と［Row］プロパティに、配列のインデックスを指定します。このプロパティで、アクセスする［Data］プロパティのインデックスを指定し、［Data］プロパティにグラフのデータを設定していきます。

8 話だけではわかりづらいと思いますので、試しに5つのデータを棒グラフにしてみましょう。新しいプロジェクトファイルを用意してください（完成版は「Chart.vbp」としてCD-ROMの「Lesson31」フォルダに収録しています）。

①ツールパレットにチャートコントロールを組み込み、フォームに配置します。チャートコントロールは、デフォルトでは4列5行のデータを棒グラフで表示します。プロパティウィンドウを見ると、［ColumnCount］が「4」に、［RowCount］が「5」になっています。

デフォルトは４列
５行の棒グラフ

②ここで、［RowCount］プロパティの値を「1」に変更します。すると、グラフは1行4列（1列はデータが0なので表示されていない）の棒グラフに変わりました。

１行４列の棒グラフ
に変更される

③フォームのLoadイベントプロシージャに、次のコードを作成します。

```
Private Sub Form_Load()

    MSChart1.Column = 1
    MSChart1.Data = 10

    MSChart1.Column = 2
    MSChart1.Data = 20

    MSChart1.Column = 3
    MSChart1.Data = 30

    MSChart1.Column = 4
    MSChart1.Data = 40

End Sub
```

そして、プログラムを起動します。グラフは、右上がりの棒グラフに変化します。

このように、一次元配列のグラフデータを設定する場合は、最初にデータの数だけ「ColumnCount」を設定して、列数を決めます。そして、1番目のデータを設定する場合は、「Column」プロパティに「1」を設定して、続けて［Data］プロパティにグラフデータを1列分だけ入力します。

```
MSChart1.Column = 1
MSChart1.Data = 10
```

2番目のデータを［Data］プロパティに設定する場合は、「Column」プロパティに「2」を設定し、［Data］プロパティにデータを代入します。こうして、グラフ作成に必要なデータを、［Data］プロパティに代入していけばグラフが作成されます。

9　このプログラムでは、2つのデータで円グラフを作成し、ディスク容量の使用量と空き容量を表示します。そのため、

プロパティ	値
ColumnCount	2
RowCount	1

という1行2列の設定をしています。

また、[ChartType] プロパティはグラフの種類を設定するプロパティで、定数でグラフタイプを指定します。[RowLabel] プロパティは、行の名前を表示するプロパティです。それぞれ、プロパティウィンドウで次の値に設定してグラフの表示を整えています。

プロパティ	値
ChartType	14 − VtChChartType2dPie
RowLabel	Disk Space

ワンポイントアドバイス

プロパティページを使ってプロパティ設定を行うと、グラフの種類・形状を一括で設定できます。

10　グラフの形式が設定できたら、[Column] プロパティで1列ずつ指定し、[Data] プロパティに値を設定していきます。ここでは、GBクラスのハードディスクの容量をグラフにするわけですから、わかりやすいグラフデータになるように、「1000000」で割った数値を、CInt関数で整数化して、MB単位でグラフ化するようにしています（例：450,560,000バイト＝約451MB）。

```
With MSChart1
    .Column = 1
    .Data = CInt(FreeTotal / 1000000)
    .Column = 2
    .Data = CInt(UsedTotal / 1000000)
End With
```

これで、[Check] ボタンがクリックされれば、ドライブの容量に関する情報をフォームに円グラフで表示します。そして、最後に [終了] ボタンにプログラムの終了処理を作成して、完成です。

```
Private Sub ExitBtn_Click()
    Unload Me
    End
End Sub
```

COLUMN ・・・・・

値渡しと参照渡し

DLLのプロシージャでは、引数は値渡しでやり取りされています。「値渡し」とは、コードを実行するのに必要な実際の値そのものを運用するやり方です。また、C言語では、配列以外のすべての引数が値渡しで渡されています。

一方、Visual Basicでは通常、特に指定しなければ「参照渡し」で引数のやり取りが行われています。参照渡しは、実際の値ではなく、値のある格納場所を運用します。

そのため、Visual Basicでの参照渡しの値をそのままDLLプロシージャに引き渡すと、プログラムが正常に動作しません。また、逆にVisual Basicで値渡しを行うと、プログラムによっては不具合を起こしてしまうものもあります。Visual Basicプログラマーは作成しているプログラムで何をしたいのかを見極め、データの渡し方を状況に応じて変更する必要があります。

ちなみに、引数を値渡しで渡すためには、Declareステートメントの引数の宣言で「ByVal」キーワードを記述します。明示的に記述しないものは自動的に参照渡しになります。たとえば、次のプロシージャでは、第1引数は参照渡し、第2、第3引数は値渡しで渡されています。

```
Declare Function InflateRect Lib "user32" _
    Alias "InflateRect" (lpRect As RECT, ByVal x As Long, _
    ByVal y As Long) As Long
```

いずれにせよ、API関数を使いこなすには、より高度なプログラミング知識が要求されます。本書ではその触りだけを、できるだけわかりやすいように解説しました。もし、APIを利用したプログラミングについてもっと知りたい場合は、他の解説書も参考にしてみてください。

まとめ

ドライブやディスクに関する情報は、Visual Basicだけでは把握できません。ファイルシステムを操作できるAPIを使用すれば、プログラムの幅も広がります。ファイラーなどのファイルユーティリティプログラムを作成するときは、これらの便利なAPIも有効に使用してください。

Win32APIには、まだまだたくさん便利な機能が揃っています。ぜひ、APIバイブルやWin32SDKなどから情報を入手して、自分のプログラミングに役立ててください。

練習問題

Q1 作成したプログラムに、指定したドライブのタイプを表示する機能を組み込んでください。

【ヒント】

API関数「GetDriveType」は、指定したドライブのタイプを返してきます。たとえば、「CDROM」とか「RAMDISK」など、ドライブの種類を定数で返してきます。

【関数】

```
Declare Function  Lib "kernel32" Alias _
    "GetDriveTypeA" (ByVal nDrive As String) As Long
```

【定数】

```
Public Const DRIVE_CDROM = 5
Public Const DRIVE_FIXED = 3
Public Const DRIVE_RAMDISK = 6
Public Const DRIVE_REMOTE = 4
Public Const DRIVE_REMOVABLE = 2
```

解答は巻末に→

Let's VBSchool!!

第10日 (Lesson32～Lesson36)
ExcelをVisual Basicで操作する

L e a r n i n g S c h o o l

Microsoft Excelでは、Excelを構成するすべての要素をVisual Basicからでも操作できるようになっています。そして、個々のオブジェクトには、プロパティとメソッド、イベントが組み込まれており、これらは、Excelのマクロ言語「Visual Basic for Application」(略して「VBA」)から、自由に操作できます。

本日のレッスンでは、主にExcelとWordを使って、Visual BasicからOfficeアプリケーションを操作するテクニックを紹介していきます。

今日のレッスンで解説する項目

❑ Excelオブジェクトのインスタンスの作成とオブジェクトへのアクセス
❑ Excelのオブジェクト構造
❑ 参照設定の方法と機能
❑ オブジェクトブラウザでExcelオブジェクトを調べる
❑ Excelのヘルプファイルの活用
❑ オブジェクト変数の設定と解放
❑ 新しいブックの追加とワークシートのセルおよびセル範囲の参照方法
❑ フォーム・コントロールからセルへのデータの代入、セルからのデータの取得
❑ オブジェクト名の取得と操作プロパティ
❑ ファイルからのExcelの起動と終了、ブックを閉じる操作と保存
❑ フォームのUnloadイベントの活用
❑ 非表示状態でのExcelの操作
❑ Errオブジェクトとそのプロパティ
❑ Windowオブジェクトの操作
❑ ワークシート関数の使用方法と活用例
❑ Rangeオブジェクトの把握と［Cells］プロパティの操作
❑ フォームとワークシートの連携操作

Lesson 32

第10日
1時限目

Excelを
Visual Basic
で操作する

Lesson32.vbp
VB6CD2
Lesson32

Excel操作の基本知識

まず、手始めにVisual BasicでExcel 97/2000（以下Excel）を操作する基本的なテクニックを身に付けます。また、Excelを構成するオブジェクトの構造についての基礎的な知識を身に付けます。Visual Basicに「Excelオブジェクト」を操作できるように設定すると、オブジェクトブラウザを使ってExcelのオブジェクトとそのプロパティ、メソッド、イベントを把握できます。また、Visual Basicのコードウィンドウから、ExcelおよびVBAのヘルプファイルを参照できます。このレッスンでは、これらのツールを有効に活用しながら、Visual BasicのプログラムからExcelを操作するコードを作成します。

今回作成するプログラム

ボタンをクリックすると…

Excel 97が起動する

POINT

フォームに配置したコマンドボタンを使ってExcelを起動し、新しいブックを追加します。そして、ワークシート「Sheet1」のセル「A1」に、「Hello Visual Basic!」という文字を挿入します。挿入した文字は、フォントサイズを18ポイントの青色にし、セル幅をその文字に合わせます。これらのExcelの操作をすべて、作成したVisual Basicのプログラムで行います。

コントロールとプロパティの設定

※新しいプロジェクトファイル（標準 EXE）を用意します。

【注意】
この 10 日目のレッスンでは、Excel（97 以降）が組み込まれていることを前提とします（本書では Excel2000 を使用します）。Excel がインストールされていない場合は、レッスンを開始する前にインストールしてください。サンプルプログラムは Excel がないと動作しません。なお、付録 CD-ROM には Excel は収録されていません。

1 フォームの［Caption］プロパティを「Excel 操作プログラム」に変更する

2 コマンドボタンコントロールを配置し、［Caption］プロパティを「Book を開く」に変更する

コマンドボタンを配置する

コマンドボタンを配置するには、ツールパレットの［CommandButton］を使います。

ラベルコントロールを配置する

ラベルコントロールを配置するには、ツールパレットの［Label］A を使います。

Excel97 の場合

Excel97 を操作する場合は、「Microsoft Excel 8.0 Object Library」をチェックします。

3 ラベルコントロールを配置し、［Caption］プロパティの文字を削除する

Excel操作プログラム
Bookを開く
配置してプロパティの文字を削除したラベルコントロール

4 Visual Basic の［プロジェクト］メニューの［参照設定］をクリックする

Lesson32 - Microsoft Visual Basic [デザイン]
ファイル(F) 編集(E) 表示(V) プロジェクト(P) 書式(O) デバッグ(D) 実行(R) クエリー(U) ダイアグラム(I) ツール(T)
フォーム モジュールの追加(F)
MDI フォーム モジュールの追加(I)
標準モジュールの追加(M)
クラス モジュールの追加(C)
ユーザー コントロールの追加(U)
プロパティ ページの追加(P)
ユーザー ドキュメントの追加(U)
DHTML Page の追加
Data Report の追加
WebClass の追加
Data Environment の追加
ファイルの追加(A)... Ctrl+D
Form1.frm の解放(R)
参照設定(N)...
コンポーネント(O)... Ctrl+T
Lesson32 のプロパティ(E)...
1 ここをクリック
2 ここをクリック

5 「Microsoft Excel 9.0 Object Library」をチェックし、［OK］ボタンをクリックする

参照設定 - Lesson32.vbp
参照可能なライブラリ ファイル(A):
Visual Basic For Applications
Visual Basic runtime objects and procedures
Visual Basic objects and procedures
OLE Automation
Microsoft Excel 9.0 Object Library
Active Setup Control Library
ActiveMovie control type library
ActiveX Conference Control
ActiveX DLL to perform Migration of MS Reposit
ATL 2.0 Type Library
CAPICHKR
CCrsWpp 1.0 Type Library
CFtpWpp 1.0 Type Library
cryptext 1.0 Type Library
OK
キャンセル
参照(B)
ヘルプ(H)
優先順位
Microsoft Excel 9.0 Object Library
場所: c:¥Program Files¥Microsoft Office¥Office¥EXCEL9.OLB
言語: 標準
1 ここをチェック
2 ここをクリック

6 フォームレイアウトウィンドウで、フォームをデスクトップの左上に移動する

フォーム レイアウト

Form1

1 ドラッグして移動

コードの記述

Form1:Command1-Click

Lesson32 - Form1 (コード)

Command1　Click

```
Private Sub Command1_Click()
    Dim AppXLS As Object

    Set AppXLS = CreateObject("excel.application.9")
    With AppXLS.Application
        .Workbooks.Add
        .Visible = True
        With .Workbooks(1).Worksheets("Sheet1").Range("A1")
            .Value = "Hello Visual Basic!"
            .Font.Size = 18
            .Font.ColorIndex = 5
            .Columns("A:A").AutoFit
        End With
    End With

    Command1.SetFocus
    MsgBox "Excelを起動しました。"
    Set AppXLS = Nothing
    Label1.Caption = "Excelの終了は、Excelで行ってください。"
End Sub
```

解　説

1　Microsoft Excel 97/2000をVisual Basicから操作する際に、操作対象のオブジェクトとして設定しておくと、オブジェクトブラウザでそのアプリケーションのオブジェクトに関する情報を表示でき、また関連するヘルプファイルを参照することができます。

Visual Basicの［プロジェクト］メニューの［参照設定］は、操作可能なオブジェクトを、Visual Basicから参照できるように設定するメニューです。このメニューを選ぶと、参照可能なオブジェクトライブラリが一覧表示されます。この一覧から、操作したいオブジェクトを選んでチェックを付ければ、フォームやコントロールと同じように、そのオブジェクトに関するプロパティやメソッド、イベントなどの構文をVisual Basicのエディタでもチェックできるようになります。

［プロジェクト］メニューの［参照設定］をクリック

操作したいオブジェクトを選び、チェックを付ける

Microsoft Excel 97/2000を参照可能なオブジェクトとして設定するには、このライブラリファイルの一覧で、Excel97は［Microsoft Excel 8.0 Object Library］に、Excel2000は［Microsoft Excel 9.0 Object Library］にチェックを付けます（Excel 97は、内部的には「バージョン8.0」、Excel2000は「バージョン9.0」として取り扱われています）。ただし、参照設定が行われるのは、設定を行った時点でアクティブになっているプロジェクトファイルのみです。したがって、複数のプロジェクトファイルでExcelオブジェクトを参照する場合は、個々のプロジェクトファイルごとに参照設定を行う必要があります。

2　オブジェクトを参照設定すると、オブジェクトブラウザの対象ライブラリにも、そのオブジェクトが組み込まれます。そのオブジェクトがどのようなオブジェクトで構成されているのか、保有しているメソッドやプロパティの種類や概要などを、オブジェクトブラウザで知ることができます。

Excelアプリケーションオブジェクトの階層構造

ワークシートのオブジェクト構造

- ワークシート（Worksheets）
  - 名前（Names）
  - セル範囲（Range）
    - エリア（Areas）
    - 罫線（Borders）
    - フォント（Font）
    - 塗りつぶし色（Interior）
    - 文字（Characters）
      - フォント（Font）
    - 名前（Names）
    - スタイル（Styles）
      - 罫線（Borders）
      - フォント（Font）
      - 塗りつぶし色（Interior）
    - 条件付き書式（FormatConditions）
    - ハイパーリンク（Hyperlinks）
    - 入力規則（Validation）
    - コメント（Comment）
    - ふりがな（Phonetic）
    - シェイプ（Shapes）
      - リンクされたOLEオブジェクト書式（LinkFormat）
      - OLEオブジェクトの書式（OLEFormat）
      - ハイパーリンク（HyperLink）
      - 塗りつぶしの書式（FillFormat）
      - コントロールの書式（ControlFormat）
      - コネクタの書式（ConnectorFormat）
      - テキスト枠（TextFrame）
      - 調整値（Adjustments）
      - 線の書式（LineFormat）
      - ピクチャの書式（PictureFormat）
      - 影の書式（ShadowFormat）
      - ワードアートの書式（TextEffectFormat）
      - HTMLスクリプトコレクション（Scripts）
  - コメント（Comment）
  - 水平改ページ（HPageBreaks）
  - 垂直改ページ（VPageBreaks）
  - ハイパーリンク（Hyperlinks）
  - シナリオ（Scenario）
  - OLEオブジェクト（OLEObjects）
  - アウトライン（Outline）
  - ページ設定（PageSetup）
  - クエリーテーブル（QueryTables）
    - パラメータ（Parameters）
  - ピボットテーブル（PivotTables）
    - ピボットテーブルのキャッシュ（PivotCache）
    - ピボットテーブルの数式（PivotFormulas）
    - ピボットテーブルフィールド（PivotFields）
      - ピボットアイテム（PivotItems）
    - OLAPキューブ（CubeFields）
  - OLEオブジェクト（OLEObjects）
  - グラフオブジェクト（ChartObjects）
    - グラフ（Chart）
    - ピボットグラフレポート内フィールド位置（PivotLayout）
  - オートフィルタ（AutoFilter）
    - オートフィルタの範囲のフィルタ（Filters）

オブジェクトのみ
オブジェクトとコレクション

3 Excelを構成するすべての要素は、オブジェクトとして扱うことができます。Excelは、Excel自身をそのオブジェクトの頂点として、階層構造状にオブジェクトを組み立てています。したがって、どのオブジェクトがどの部分を構成しているのか、どのようなプロパティやメソッドを持っているのかを知らなければ、Excelを操作できません。

オブジェクトブラウザでExcelを対象オブジェクトに設定すると、操作可能なオブジェクトが［クラス］に一覧表示されます。たとえば、［クラス］から［Worksheet］を選ぶと、Worksheetオブジェクトが持っているメソッド、プロパティ、イベントが、右のウィンドウに一覧表示されます。

そして、知りたい項目をクリックして［？］ボタンをクリックすれば、その項目に関するヘルプを表示できます。

4 HTML形式のヘルプでは、左側に目次を表示し、ここから知りたい情報を調べることもできます。

ここを選択

新たに表示された目次から、[Microsoft Excel Visual Basic リファレンス]を選べば、Excelに関するVisual Basicのリファレンスを読むことができます。「検索」タブを使えば、キーワードでの検索も可能です。

単語でヘルプを検索できる

ワンポイントアドバイス

Excelには、Excelの操作を自動化するマクロ機能を搭載しています。そして、このマクロは、Visual Basicをベースに、アプリケーションのマクロ用に改修された言語「Visual Basic for Applications」(VBA)で作成されます。

VBAは、Visual Basicとほぼ同様のステートメント(Ifステートメントや For...Nextステートメントなど)で処理を制御でき、文字列処理や日付時刻の関数も、Visual Basicと同様のものが使用できます。

VBAの詳しい使い方については、小社から『10日でおぼえるExcel VBA 入門教室2000対応』が出版されていますので、ぜひそちらもご利用ください。

5 　では、実際にプログラムを作成しながら、Excelを操作する方法について理解を深めていきましょう。このプログラムは、プログラムからExcelを起動し、ワークシートに文字を挿入します。

Excelのすべての操作をプログラムのコードから行います。なお、このプログラムでは、Excelの終了はExcel側の終了操作で行うようにします。

①フォームに配置したコマンドボタンをクリックすると、Excelのインスタンスを作成します。
②Excelに新しいブックを追加し、表示します。
③セル「A1」に、文字列「Hello Visual Basic!」を挿入します。
④その文字のフォントサイズを「18」に、文字色を青色に変更します。
⑤セルの幅を、文字列の幅に合わせます。
⑥フォームに配置したラベルコントロールに、メッセージを表示します。

6 　処理はすべて、コマンドボタンのイベントプロシージャで行います。
最初に、オブジェクト変数「AppXLS」を宣言しておきます。

```
Private Sub Command1_Click()
    Dim AppXLS As Object
```

ExcelをVisual Basicから使用するには、まず「CreateObject」関数を使って、そのオブジェクトのインスタンスを生成します。「CreateObject」関数の引数には、作成するインスタンスの元になるアプリケーション名を指定します。

この関数は、第7日の「Lesson 22　インターネットエクスプローラを操作する」で使用しています。Lesson 22では、「CreateObject」関数の引数には「InternetExplorer.application」を指定しましたが、Excel2000の場合は、「excel.application.9」と、バージョン番号まで指定します（Excel97は「8」）。

「CreateObject」関数がインスタンスの作成に成功すると、そのオブジェクト名を返してきますので、Setステートメントでオブジェクト変数「AppXLS」に格納しておきます。このオブジェクト変数は、あとでExcelの各オブジェクトを操作する際に使用します。

```
Set AppXLS = CreateObject("excel.application.9")
```

7 　Excelのインスタンスを作成しても、このままではワークシートは操作できません。Excelに新しいブックを追加する必要があります。ブックの追加は、「Workbooks」オブジェクトを指定して「Add」メソッドを実行します。

ExcelのVBAでは［Workbooks］プロパティを使用すると、「Workbooks」オブジェクトを指定することができます。ややこしい話になりますが、この辺が、Excel独特のオブジェクト構造とプロパティの仕組みになっていて、つまり、オブ

ジェクトの名称と同じ名前のプロパティを使うと、そのオブジェクトが取得できるようになっています。

また、ブックやワークシート、セルなどのように、同じオブジェクトがいくつも集まっているものを、「コレクションオブジェクト」と呼んでいます。たとえば、「Worksheet」オブジェクトはワークシート単体を指しますが、いくつもあるワークシートをまとめて「Worksheets」コレクションオブジェクトとして扱います。

ブックを追加する場合は、「Workbooks」コレクションオブジェクトを指定して、「Add」メソッドを実行します。このとき、親のオブジェクトに、インスタンスで作成したExcelオブジェクトを指定しておきます。1つのブックでも「Workbooks」になります。

```
With AppXLS.Application
    .Workbooks.Add
```

ブックを追加したら、Excelオブジェクトの［Visible］プロパティに「True」を設定すると、Excelと新規に追加したブックが表示されます。

```
.Visible = True
```

ブックが新規に追加されて表示される

この［Visible］プロパティと［Workbooks］プロパティは、Excelでは「Application」オブジェクトに属するプロパティになります。「Application」オブジェクトとは、Excel自身を指すオブジェクトのことで、このオブジェクトが所有するプロパティを使用すれば、Excelの起動や終了、表示形態、表示サイズなどを指定できます。通常、Excel自身を操作する場合は、「Application」オブジェクトを対象オブジェクトに指定して、プロパティやメソッドを操作することになります。

オブジェクトブラウザで調べたExcelの「Application」オブジェクトのプロパティやメソッド

8 　ブックを追加して表示したら、ワークシート「Sheet1」に文字を挿入します。ワークシートは、「Workbook」オブジェクトの中の「WorkSheets」コレクションオブジェクトとして扱われます。

　ブックには、何枚ものワークシートが組み込まれており、グラフシートなども組み込まれています。これらは、1つ1つのオブジェクトになっており、単体では「WorkSheet」オブジェクトとして、ワークシート全体では「WorkSheets」コレクションオブジェクトとして扱うことができます。

「Workbook」オブジェクトの［WorkSheets］プロパティを使用すると、単体の「WorkSheet」オブジェクトを取得できます。このプロパティは、引数に取得したいワークシート名またはインデックス番号を指定します。

　Excelオブジェクト「AppXLS」のワークシート「Sheet1」にアクセスしたい場合は、次のように記述します。

```
AppXLS.Workbooks(1).Worksheets("Sheet1")
```

　このとき、どのブックのワークシートなのかをはっきりさせる必要がありますから、ブックのオブジェクト指定を［Workbooks］プロパティで指定する必要があります。［Workbooks］プロパティは、ブックの名前またはインデックス番号を引数に指定するとそのオブジェクトを参照できます。

　ブックの名前がわからない場合は、番号でブックを指定すると便利です。ここでは、新しいブックを1つ追加しただけなので、「Workbooks(1)」と記述すれば、そのブックを指定できます。

9 今度は、ワークシート内のセル「A1」を指定して、文字をセルに代入します。セルは「Range」というオブジェクトで扱います。このオブジェクトを取得・参照するには、[Range] プロパティを使用します。引数に指定したいセル番地を記述します。

```
Range("A1")
```

これで、やっとセル「A1」を指定できました。このように、Excelではセル1つを取ってみても、同じようなものがいくつも存在していますから、どのオブジェクトに属するどのオブジェクトなのか、を階層状に指定していかなければなりません。

このプログラムでは、Excelのインスタンスであるオブジェクト「AppXLS」の1番目のブックの、ワークシート「Sheet1」のセル「A1」を操作することになりますから、これらのオブジェクト名を順番につなげて記述する必要があります。

```
AppXLS.Workbooks(1).Worksheets("Sheet1").Range("A1")
```

オブジェクト「AppXLS」の1番目のブックのワークシート「Sheet1」のセル「A1」

10 セルに値を代入するには、「Range」オブジェクトの「Value」プロパティにその値を代入します。文字列の場合は、"" (ダブルクォーテーション) で囲んで記述し、数値の場合は直接数字を代入します。

```
.Value = "Hello Visual Basic!"
```

セルのフォントサイズを変更するには、「Font」オブジェクトの［Size］プロパティにサイズを数値で代入します。また、文字色を変更したい場合は、「Font」オブジェクトの［ColorIndex］プロパティに色番号を設定します。

```
.Font.Size = 18
.Font.ColorIndex = 5
```

Excelでは、文字属性はすべて「Font」オブジェクトとして扱われます。Excelで使われる文字は、セルだけではありません。グラフでもコメントでも図形でも使用されます。したがって、フォントだけを1つにまとめて独立したオブジェクトとして取り扱うようにし、どのオブジェクトのフォントを操作するのか、親のオブジェクトをいっしょに指定するようになっています。

Fontオブジェクトには、いくつものプロパティが用意されており、フォントに関するいろいろな設定が行えます。

11

Excelの操作の最後は、セル幅を文字の長さにそろえる操作です。Rangeオブジェクトの［Columns］プロパティを使うと、引数に指定した列範囲すべてをオブジェクトとして把握できます。また、Rangeオブジェクトの「AutoFit」メソッドは、指定したセルの幅や高さを、セルの内容に合わせてくれます。必ず列または行全体を指定しますので、セル幅の場合は［Columns］プロパティと組み合わせて実行します。

```
.Columns("A:A").AutoFit
```

ワンポイントアドバイス

セルの高さを内容に合わせる場合は、［Rows］プロパティと組み合わせます。また、これらの［Columns］、［Rows］はVisual Basicのリストボックスコントロールのスクロール方向を決めるときにも同名のプロパティが使用されますから、混同しないように注意してください。

12

ここで、一度フォームにフォーカスを戻します。もしフォーカスを戻さないでおくと、フォーカスはExcelに移動したままになります（Excelのウィンドウが最前面に表示された状態）。

そしてその後、メッセージボックスでExcelを起動したことを知らせます。

```
Command1.SetFocus
MsgBox "Excelを起動しました。"
```

Excelの終了は、Excelそのものの操作で終了するようにしますので、プログラムとしては作成したインスタンスのオブジェクト変数を開放しておきます。これは、Setステートメントでオブジェクト変数に「Nothing」を代入します。

```
Set AppXLS = Nothing
```

最後にラベルコントロールにメッセージを表示して終了です。

```
Label1.Caption = "Excelの終了は、Excelで行ってください。"
```

まとめ

Excel を操作するには、まず「CreateObject」関数でオブジェクトのインスタンスを作成します。このとき、必ずインスタンスのオブジェクト名をオブジェクト変数に格納してください。作成した Excel のインスタンスを操作する際に、このオブジェクト変数を使って、Excel オブジェクトを指定するからです。

Excel の個々の要素は、すべてオブジェクトとして扱います。したがって、Excel を構成するオブジェクトの階層と概要をよく把握していないとなにも操作できません。Excel のオブジェクト構造は、階層状になっており少し複雑です（P.421 〜 422 参照）。Excel のヘルプを良く調べて、オブジェクトと保有するプロパティやメソッドを把握しておいてください。

練習問題

Q1　Excel が起動するときに、Excel の幅を「700」、高さを「500」でデスクトップの左上に表示されるように、作成したプログラムを改修してください。

【ヒント】

[Width]、[Height]、[Left]、[Top] の各プロパティを使用します。

Excel をサイズとデスクトップの表示位置指定で起動する

解答は巻末に→

Lesson 33

第10日 2時限目

Excelを Visual Basic で操作する

Lesson33.vbp

VB6CD2

Lesson33

ファイルからオブジェクトを作成する

前のレッスンでは、新しいExcelオブジェクトをインスタンスとして作成しました。今度は、起動したいアプリケーションのデータファイルがあらかじめ存在する場合に、直接アプリケーションを起動してファイルを表示する方法を紹介します。「GetObject」関数は、引数にデータファイル名を指定すれば、そのままアプリケーションを起動してファイルを表示できます。この関数を使用して、直接ファイル名を指定してExcelを起動するプログラムを作成します。

今回作成するプログラム

ここをクリックすると、新しいExcelが起動して指定したブックが開く

新しいブックを起動

ファイルからExcelを起動

ここをクリックすると、Excelが既に起動していたら、そのExcelに指定したファイル開き、Excelが起動していなかったら、Excelが起動し、ファイルが開く

POINT

2つのコマンドボタンを使用して、1つは前のレッスンで行ったように、「CreateObject」関数でExcelを起動し、新しいブックを追加します。もう1つは、「GetObject」関数を使用して、Excelのデータファイルを指定してExcelを起動します。また、いずれのボタンも、Excelを起動した後に、別のブックを開く操作を実行します。

コントロールとプロパティの設定

※新しいプロジェクトファイル（標準EXE）を用意します。また、付録CD-ROMの「Lesson33」フォルダにあるExcelのブック「photoalbum.XLS」と「野菜栄養価.XLS」を、プログラムを作成するフォルダにコピーしてください。

1 フォームにコマンドボタンを2つ配置し、各コントロールの［Caption］プロパティを設定する

コントロール	［Caption］プロパティの値
CommandButton1	新しいブックを起動
CommandButton2	ファイルからExcelを起動

配置してプロパティを設定したコマンドボタンコントロール

コマンドボタンを配置する

コマンドボタンを配置するには、ツールパレットの［CommandButton］を使います。

2 ［プロジェクト］メニューの［参照設定］を選び、［Microsoft Excel 9.0 Object Library］を選ぶ

1 ここをチェック

2 ここをクリック

Excel97の場合

［Microsoft Excel 8.0 Object Library］を選んでください。

コードの記述

Form1:(General)-(Declarations)

Lesson33 - Form1 (コード)

(General) | (Declarations)

```
Dim XLSBook As Object
```

Form1:Command1-Click

Lesson33 - Form1 (コード)

Command1 | Click

```
Private Sub Command1_Click()
    Dim Fname As String

    On Error GoTo FAIL
    Set XLSBook = CreateObject("excel.application.9")
    MsgBox "新しくExcelを起動しました。"
    With XLSBook.Application
        .Visible = True
        Fname = App.Path & "" & "photoalbum.XLS"
        .Workbooks.Open (Fname)
    End With
    Set XLSBook = Nothing
    Exit Sub

FAIL:
    MsgBox Error(Err.Number)
End Sub
```

Form1:Command2-Click

Lesson33 - Form1 (コード)

Command2 | Click

```
Private Sub Command2_Click()
    Dim Fname As String

    Fname = App.Path & "" & "野菜栄養価.XLS"
    On Error GoTo FAIL
    Set XLSBook = GetObject(Fname)
    MsgBox "ファイルからExcelを起動しました。"
    With XLSBook.Application
        .Visible = True
        .Windows("野菜栄養価.XLS").Visible = True
        Fname = App.Path & "" & "photoalbum.XLS"
        .Workbooks.Open (Fname)
```

```
    End With
    Set XLSBook = Nothing
    Exit Sub

FAIL:
    MsgBox Error(Err.Number)
End Sub
```

解 説

1 　このプログラムは、2つの方法でExcelを起動します。最初のコマンドボタン［新しいブックを起動］は、すでに前のレッスンで使用した「CreateObject」関数を使用してExcelを起動します。もう1つのボタン［ファイルからExcelを起動］は、「GetObject」関数で既存のブックを指定してExcelを起動します。

「GetObject」関数は、引数にファイルを指定すると、そのファイルに関連付けられているアプリケーションを起動し、指定したファイルを表示します。すでに起動したいブック名がわかっている場合は、「GetObject」関数で直接ブックを開いたほうが処理が簡単です。

「GetObject」関数は、アプリケーションの起動に成功すると、そのオブジェクト名を返してきます。存在しないファイル名を指定した場合など、アプリケーションの起動に失敗すると「GetObject」関数はエラーを発生します。

2 　では、実際にコードを作成して、それぞれの関数の違いを見てみましょう。まず、フォームの宣言セクションに、オブジェクト変数「XLSBook」を宣言します。次に、「新しいブックを起動」ボタンのClickイベントプロシージャに、「CreateObject」関数でExcelを起動するコードを記述します。「CreateObject」関数を、プログラムID「excel.application.9」を引数にして実行します。戻り値は、オブジェクト変数「XLSBook」に格納します。そして、このExcelオブジェクトと［Application］プロパティでオブジェクトを指定して、［Visible］プロパティを「True」に設定します。ここまでの処理では新しいブックを追加していませんから、ブックが何もないExcelが起動します。

```
Private Sub Command1_Click()
    Dim Fname As String

    On Error GoTo FAIL
    Set XLSBook = CreateObject("excel.application.9")
    MsgBox "新しくExcelを起動しました。"
    With XLSBook.Application
        .Visible = True
```

空のExcelが起動する

3 今度は、プロシージャの先頭で宣言していた変数「Fname」に、ファイル名を作成して格納します。「App」オブジェクトの［Path］プロパティを使うと、現在プログラムを実行しているディレクトリのパス名を把握できます。このプログラムと同じフォルダには、Excelのブック「photoalbum.XLS」が保存されていますから、この［Path］プロパティと組み合わせて、フルパスのファイル名を作成します。

```
Fname = App.Path & "" & "photoalbum.XLS"
```

次に、Excelの［Workbooks］プロパティで「Workbooks」オブジェクトを指定して、Excel VBAのOpenメソッドを実行します。Visual BasicのOpenメソッドは、シーケンシャルやバイナリ、ランダムアクセスファイルを開くのに使用しますが、Excel VBAのOpenメソッドは、対象オブジェクトに「Workbooks」オブジェクトを指定すると、ブックを開きます。ここでは、変数「Fname」に格納した、「photoalbum.XLS」をフルパスで設定します。これではじめて、Excelにブックが表示されます。

```
.Workbooks.Open (Fname)
```

Excelのブックが開く

ブックが開いたら、Setステートメントでオブジェクト変数に「Nothing」を代入して、変数を解放します。

```
Set XLSBook = Nothing
Exit Sub
```

4　「CreateObject」関数は、オブジェクトの作成に失敗するとエラーを発生します。たとえば、プログラムを実行しようとしたコンピュータに、Excelがインストールされていなかったり、プログラムIDの指定を間違えた、というときはエラーを発生してプログラムを停止します。

これを回避するために、「CreateObject」関数の前に「On Error GoTo」ステートメントを設定し、ラベル「FAIL」に処理をジャンプさせます。ラベル「FAIL」では、Error関数とErrオブジェクトの［Number］プロパティを使用して、発生したエラーの番号に合ったエラーメッセージを表示します。

```
FAIL:
    MsgBox Error(Err.Number)
```

ここでは、次のようなエラーメッセージが発生します。

Lesson33

ActiveX コンポーネントはオブジェクトを作成できません。

OK

5　次に、［ファイルからExcelを起動］ボタンの処理を作成します。このボタンでは、「GetObject」関数を使って、直接ブックを開きます。「GetObject」関数の引数に、ブックのファイル名を設定して実行すると、そのファイルに関連付けられているアプリケーションが起動し、ファイルを表示してくれます。

最初に、オブジェクト変数を宣言セクションで宣言し、［ファイルからExcelを起動］ボタンのClickイベントプロシージャで、起動するブックのファイル名をフルパスで作成します。

```
Dim XLSBook As Object

Private Sub Command2_Click()
    Dim Fname As String

    Fname = App.Path & "" & "野菜栄養価.XLS"
```

次に、「GetObject」関数を実行します。この関数は成功すると作成したオブジェクト名を返してきますから、オブジェクト変数「XLSBook」に格納します。また、ファイル名が違うなど存在しないファイルを開こうとした場合は、エラーを発生してプログラムが停止してしまいますから、「On Error GoTo」ステートメントで、ラベル「FAIL」にジャンプさせます。

```
On Error GoTo FAIL
Set XLSBook = GetObject(Fname)
MsgBox "ファイルからExcelを起動しました。"
```

これで、Excelが起動しますが、この時点ではまだブック「野菜栄養価.XLS」は表示されません。

6 次に、Applicationオブジェクトを指定して、[Visible] プロパティに「True」を設定します。これではじめてExcelが画面に表示されます。しかし、まだブックは表示されません。

```
With XLSBook.Application
     .Visible = True
```

そこで、Applicationオブジェクトの「Window」オブジェクトにある [Visible] プロパティに「True」を設定します。「Window」オブジェクトは、アプリケーションのすべてのウィンドウを1つにまとめたオブジェクトで、「Windows」コレクションオブジェクトとしても扱います。単一のWindowオブジェクトにアクセスするには「Windows()」プロパティを使用します。

指定方法は、「Windows(1)」のようにインデックス番号かファイル名を指定することでオブジェクトを取得できます。Excelは起動マクロファイルなどなにか開いている可能性もありますので、ここでは「Windows("野菜栄養価.XLS")」と直接オブジェクトを指定してみます。

[Visible] プロパティは、ウィンドウを表示するかどうかを設定するプロパティです。そこでブック「野菜栄養価.XLS」のプロパティを「True」にして表示状態にします。

```
.Windows("野菜栄養価.XLS").Visible = True
```

「野菜栄養価.XLS」が表示される

7 引き続き、Excel VBAのOpenメソッドでブック「photoalbum.XLS」を読み込みます。Openメソッドは、読み込んだ時点でブックを表示しますから、特に [Visible] プロパティの操作は必要ありません。

```
Fname = App.Path & "" & "photoalbum.XLS"
.Workbooks.Open (Fname)
```

	A	B	C	D	E
1	写真名	ファイル名	カメラ	レンズ	撮影日
2	ロビー	airport.jpg	PENTAX Z-50P	28-80mm Power Zoom	97年8月
3	時計の建物	clock.jpg	Canon A1	135mm	96年7月
4	馬と子供	hose.jpg	Canon A1	85mm	96年8月
5	バラープレイボーイ	rose-1.jpg	Canon A1	85mm + クローズアップフィルタ	95年6月
6	窓と花	window.jpg	PENTAX Z-50P	28-80mm Power Zoom	97年8月

「photoalbum.XLS」が表示される

そして、オブジェクト変数の解放とエラー処理を作成して完成です。

```
    Set XLSBook = Nothing
    Exit Sub

FAIL:
    MsgBox Error(Err.Number)
End Sub
```

まとめ

「CreateObject」関数は、新しいExcelオブジェクトのインスタンスを作成します。「GetObject」関数は、指定したファイルとそのアプリケーションを起動する関数です。

どちらの関数を使用しても、Addメソッドによる新しいブックの追加や、Openメソッドで指定したブックを表示できます。

また、「CreateObject」関数は、いくつでもインスタンスを作成できますが、「GetObject」関数は同じファイルはいくつも開けません。目的に合わせて、これらの関数を使い分けてください。

練習問題

Q1　プログラムにコマンドボタンを1つ追加し、現在表示しているウィンドウ名を、メッセージボックスで表示するように修正してください。

解答は巻末に→

Lesson 34

第10日 3時限目

Excelを Visual Basicで操作する

Lesson34.vbp
VB6CD2
Lesson34

ワークシート関数を使う

Excelの機能の大きな特徴の1つに、豊富なワークシート関数をもっていることがあげられます。算術関数や統計関数、財務関数など、いちいち式を組み立てていたのでは大変な計算も、これらのワークシート関数を使用すればすぐに答えを導き出せます。そこで、このレッスンでは、Visual Basicのプログラムから、Excelのワークシート関数を使用するテクニックを紹介します。

今回作成するプログラム

計算プログラム － Excelを起動中です

入力

10000
20000
30000
40000
50000

入力したデータを一覧表示する

Max 50000

入力したデータの最大値を表示する

Ave 30000

入力したデータの平均値を表示する

終了

POINT

フォームに配置したテキストボックスコントロールに数字を入力すると、非表示状態で起動したExcelのワークシートにその値を代入します。そして、入力されたセルデータを元に、ワークシート関数のMax関数で最大値を、Average関数で平均値を算出し、フォームに表示します。Visual Basicのコードから、Excelのセル範囲を把握してワークシート関数を実行させるのがポイントです。

コントロールとプロパティの設定

※新しいプロジェクトファイル（標準EXE）を用意します。また、付録CD-ROMにある「material」フォルダのアイコンファイル「CALC.ICO」を、使用しているコンピュータにコピーしてください。

1 ［プロジェクト］メニューの［参照設定］を選び、［Microsoft Excel 9.0 Object Library］を選ぶ

2 フォームの［Caption］プロパティを「計算プログラム」に変更する

Excel97の場合

［Microsoft Excel 8.0 Object Library］を選んでください。

[...] ボタン

[...] ボタンは、そのプロパティの詳細な指定を行うダイアログボックスが用意されていることを示しています。

テキストボックスを配置する

テキストボックスコントロールを配置するには、ツールパレットの[TextBox] abl を使います。

3 [Icon] プロパティの [...] ボタンをクリックし、「CALC.ICO」を設定する

4 テキストボックスコントロールを配置し、次のプロパティを設定する

プロパティ	値
Text	(なし)
TabIndex	0

5 ツールパレットの [ListBox] を使って、リストボックスコントロールを配置する

6 コマンドボタンコントロールを４つ配置し、各コントロールのプロパティを変更する

コントロール	オブジェクト名	Caption	Enabled
Command1	InputBtn	入力	True
Command2	MaxBtn	Max	False
Command3	AveBtn	Ave	False
Command4	ExitBtn	終了	True

7 ラベルコントロールを２つ配置し、次のプロパティを変更する

プロパティ	値
BorderStyle	1 ー実線
Caption	(なし)

コマンドボタンを配置する

コマンドボタンを配置するには、ツールパレットの［CommandButton］を使います。

ラベルコントロールを配置する

ラベルコントロールを配置するには、ツールパレットの［Label］ A を使います。

コントロールをコピーする

プロパティの設定が同じコントロールを複数作成する場合は、コントロールを1つ作成したら、それをコピーして貼り付けると効率的に作業できます。

コードの記述

Form1:(General)-(Declarations)

Lesson34 - Form1 (ｺｰﾄﾞ)

(General) | (Declarations)

```
Dim AppXLS As Object
Dim i As Integer
```

Form1:Form-Load

Lesson34 - Form1 (ｺｰﾄﾞ)

Form | Load

```
Private Sub Form_Load()
    On Error GoTo FAIL
    Set AppXLS = CreateObject("Excel.Application")
    AppXLS.Workbooks.Add

    With Form1
        .Caption = .Caption & "  -  Excelを起動中です"
    End With
    i = 0
    Exit Sub

FAIL:
    MsgBox Error(Err.Number)
End Sub
```

Form1:InputBtn-Click

Lesson34 - Form1 (ｺｰﾄﾞ)

InputBtn | Click

```
Private Sub InputBtn_Click()
    i = i + 1

    If Text1.Text = "" Then
        Text1.SetFocus
        Exit Sub
    End If
    List1.AddItem Text1.Text

    With AppXLS.Workbooks(1).Worksheets("Sheet1")
        .Cells(i, 1) = Text1.Text
    End With

    Text1.Text = ""
```

```
    Text1.SetFocus

    MaxBtn.Enabled = True
    AveBtn.Enabled = True
End Sub
```

Form1:MaxBtn-Click

Lesson34 - Form1 (コード)

MaxBtn　Click

```
Private Sub MaxBtn_Click()
    Dim CheckRange As Range

    On Error Resume Next
    Set CheckRange = AppXLS.Workbooks(1) _
        .Worksheets("Sheet1").Range(Cells(1, 1), Cells(i, 1))

    Label1(0).Caption = AppXLS.WorksheetFunction.Max(CheckRange)
    Text1.SetFocus
End Sub
```

Form1:AveBtn-Click

Lesson34 - Form1 (コード)

AveBtn　Click

```
Private Sub AveBtn_Click()
    Dim CheckRange As Range

    On Error Resume Next
    Set CheckRange = AppXLS.Workbooks(1) _
        .Worksheets("Sheet1").Range(Cells(1, 1), Cells(i, 1))

    Label1(1).Caption = _
        AppXLS.WorksheetFunction.Average(CheckRange)
    Text1.SetFocus
End Sub
```

Form1:ExitBtn-Click

Lesson34 - Form1 (コード)

ExitBtn　Click

```
Private Sub ExitBtn_Click()
    Unload Me
    End
End Sub
```

Form1:Form-Unload

Lesson34 - Form1 (コード)

Form ▼ | Unload ▼

```
Private Sub Form_Unload(Cancel As Integer)
    AppXLS.Workbooks(1).Close SaveChanges:=False
    AppXLS.Quit

    Set AppXLS = Nothing
End Sub
```

解 説

1 このプログラムは、次のように動作します。

①フォーム起動時に、CreateObject関数でExcelオブジェクトのインスタンスを作成します。そして、新しいブックを追加します。Excelは非表示状態で操作します。

②フォームに配置したテキストボックスコントロールに数字を入力し、[入力] ボタンをクリックします。入力内容は、リストボックスコントロールに転送されると同時に、ワークシートのセルにも転送されます。

③データが入力されると、[Max] と [Ave] ボタンが有効になります。

④ [Max] ボタンと [Ave] ボタンをクリックすると、入力されたデータのセル範囲を把握して、ワークシート関数の「Max」と「Average」を実行し、フォームのラベルコントロールに値を表示します。

⑤プログラムを終了すると、フォームのUnloadイベントプロシージャで、Excelを終了させます。このとき、ブックは保存せずに終了します。そして、オブジェクト変数を解放します。

⑥Excelは、起動しても非表示状態にし、ユーザーが操作できないようにして、ワークシート関数の実行をします。Excelの操作はすべてプログラム側で行います。

2 では、実際にコードを作成しましょう。まず、フォームの宣言セクションに、オブジェクト変数とセル番地格納用の変数を宣言します。

```
Dim AppXLS As Object
Dim i As Integer
```

続いて、フォームのLoadイベントプロシージャで、CreateObject関数を使って、Excelのインスタンスを作成します。この時、関数の引数にExcelのバージョン番号を指定せずに、関数を実行します。フォーム起動時の処理のため、Excelのインスタンスは1つしか作成されません。作成したら、新しいブックを追加します。

```
Private Sub Form_Load()
    On Error GoTo FAIL
    Set AppXLS = CreateObject("Excel.Application")
    AppXLS.Workbooks.Add
```

Excelの［Visible］プロパティを設定しませんので、Excelはユーザーには非表示の状態で起動します。そのために、Excelが無事起動していることを、フォームのキャプションに表示します。

```
With Form1
    .Caption = .Caption & "  -  Excelを起動中です"
End With
```

また、変数「i」を「0」に初期化しておきます。

```
i = 0
```

最後に、CreateObject関数のエラー処理を作成します。

```
FAIL:
    MsgBox Error(Err.Number)
End Sub
```

3 フォームに配置した［入力］ボタンをクリックすると、テキストボックスに入力されたデータを、リストボックスコントロールとワークシートのセルに転送します。
まず、変数「i」に「1」を加えます。この変数は、［Cells］プロパティでセル番地を参照する際の、列方向のセル番地を格納するのに使います。最初は、セル「1,1」（A1）に入力しますので、「1」を加えます。

```
Private Sub InputBtn_Click()
    i = i + 1
```

次に、テキストボックスコントロールにデータがあるかどうかをチェックします。もし、［Text］プロパティの値が空なら、以降のデータ転送処理をキャンセルし、プロシージャを終了させます。その際に、テキストボックスコントロールにフォーカスを移し、次の入力に備えます。

```
If Text1.Text = "" Then
    Text1.SetFocus
    Exit Sub
End If
```

［Text］プロパティに値があれば、その値をリストボックスコントロールに、リスト項目として追加します。

```
List1.AddItem Text1.Text
```

また、ワークシートのセルにも、［Text］プロパティの値を代入します。

```
With AppXLS.Workbooks(1).Worksheets("Sheet1")
    .Cells(i, 1) = Text1.Text
End With
```

テキストボックスの内容をリストボックスとセルに転送する

実際は見えていない

ワンポイントアドバイス

テキストボックスコントロールに入力されたデータは、文字列として扱われますが、Excelはセルに数字が入力されると、それが文字列であっても数値データとして格納します。

［Text］プロパティの値の転送が終わったら、［Text］プロパティをクリアにし、フォーカスをテキストボックスに移して、次の入力に備えます。

```
Text1.Text = ""
Text1.SetFocus
```

また、この時点でデータがセルに代入されたので、[Max]と[Ave]ボタンを有効にします。

```
MaxBtn.Enabled = True
AveBtn.Enabled = True
```

ワンポイントアドバイス

当然、リストの数値は2つ以上入力してください。セルにデータがないのに、「Max」関数や「Average」関数を実行するとエラーになってしまいます。セルにデータが入力された時点で、これらの関数の実行が可能になるような処置が必要です。

4 データが転送されたら、[Max]ボタンが有効になります。このボタンは、クリックされた時点で、ワークシートのセル範囲を把握し、Rangeオブジェクトとしてオブジェクト変数「CheckRange」に格納します。変数「i」には最後に転送したセルの、列方向のセル番地が格納されているので、データが入力されているセルは、セル番地「1,1」から「i,1」までです。これを使って、データの入力されているセル範囲を把握し、オブジェクト変数「CheckRange」に、Setステートメントで格納します。

```
Private Sub MaxBtn_Click()
    Dim CheckRange As Range

    On Error Resume Next
    Set CheckRange = AppXLS.Workbooks(1) _
      .Worksheets("Sheet1").Range(Cells(1, 1), Cells(i, 1))
```

ワンポイントアドバイス

[Range]プロパティは、「Range("A1:A7")」のようにして、セル範囲を指定できますが、一方では[Cells]プロパティと組み合わせて、「Range(Cells(1, 1), Cells(i, 1))」という使い方もできます。変数でセル範囲を求める場合は、この使い方のほうが便利です。

データの入力されているセル範囲を把握したら、ワークシート関数を実行します。ワークシート関数をVisual Basic(またはVBA)で使用する場合は、「Worksheet

Function」オブジェクトの関数として使用します。たとえば、Max関数を使う場合は、以下のように記述します。

```
AppXLS.WorksheetFunction.Max(CheckRange)
```

Max関数は、引数にセル範囲を指定すると、その範囲のデータの中で最も大きい値を返してくる関数です。このとき注意するのは、Max関数の引数に設定するセル範囲は、Rangeオブジェクトとして指定する必要がある、ということです。そのために、事前にSetステートメントでセル範囲をRangeオブジェクト型の変数に設定しています。Max関数の戻り値は、そのままラベルコントロールの［Caption］プロパティに設定します。

```
Label1(0).Caption = AppXLS.WorksheetFunction.Max(CheckRange)
```

そして、テキストボックスコントロールにフォーカスを移し、次の入力に備えます。

```
Text1.SetFocus
```

5

［Ave］ボタンの処理も、基本的には［Max］ボタンの処理と同じです。使用する関数が「Average」関数になっているだけです。Average関数は、引数にセル範囲を指定すると、その範囲内のデータの平均値を返してくる関数です。Max関数と同様、引数のセル範囲には、Rangeオブジェクトを指定します。

関数を実行したら、もう1つのラベルコントロールに出力し、テキストボックスにフォーカスを移します。

```
Private Sub AveBtn_Click()
    Dim CheckRange As Range

    Set CheckRange = AppXLS.Workbooks(1) _
      .Worksheets("Sheet1").Range(Cells(1, 1), Cells(i, 1))

    Label1(1).Caption = AppXLS.WorksheetFunction _
       .Average(CheckRange)
    Text1.SetFocus
End Sub
```

6

最後に、プログラムの終了処理を作成します。ここでのポイントは、フォームのUnloadイベントプロシージャで、Excelの終了処理を行っている点です。Excelは、開いているブックを、ExcelのCloseメソッドで閉じています。このとき、名前付き引数「SaveChanges」に、値「False」を設定すると、ブックの変更を保存せずにそのまま終了します。

このプログラムでは、Excelはただ単にワークシート関数を使うだけのプロセスとして使用しただけなので、ブックは保存せずに閉じています。そして、ExcelのQuitメソッドで、Excel自身を終了させ、オブジェクト変数をSetステートメントと「Nothing」で解放しておきます。

```
rivate Sub ExitBtn_Click()
    Unload Me
    End
End Sub

Private Sub Form_Unload(Cancel As Integer)
    AppXLS.Workbooks(1).Close SaveChanges:=False
    AppXLS.Quit

    Set AppXLS = Nothing
End Sub
```

まとめ

Excelの、高機能なワークシート関数を使用すれば、VBのプログラム側で複雑な計算式を組み立てなくてすみます。Excel VBAで作成できないインターフェイスや機能を持たせたプログラムの中で、難しい計算式を組み立てる場合は、Excelのワークシート関数を利用するのも一つの手段でしょう。

ワークシート関数を使用する場合は、「WorksheetFunction」オブジェクトの関数として実行することと、引数のセル範囲はRangeオブジェクトで指定する、という点に注意してください。各ワークシート関数の詳しい使い方は、Excelのヘルプを参照してください。また、巻末の付録「Visual Basicで使用できるワークシート関数一覧」も参照してください。

練習問題

Q1　作成したプログラムに、ワークシート関数「Min」を使用して、最小値を求める機能を組み込んでください。また、Enterキーを押してもリストに入力できるように改良してみましょう。

解答は巻末に→

Lesson 35

第10日
4時限目

Excelを
Visual Basic
で操作する

Lesson35.vbp
VB6CD2
Lesson35

為替換算プログラムの作成

もう1つ、ワークシート関数を使用したプログラムを作成します。このプログラムは、ドルー円を換算するプログラムです。簡単な割り算を実行するプログラムですが、ドル表記を作成するのに、Excelのワークシート関数を使用します。このプログラムでは、Excelのインスタンスオブジェクトを作成しますが、ブックは追加せずにワークシート関数だけを使うプログラムです。

今回作成するプログラム

POINT

非表示状態でExcelのオブジェクトを作成しますが、ブックやワークシート、セルは一切使用せずに、ワークシート関数だけを利用します。Excelを利用することでプログラムを非常にコンパクトにできます。

コントロールとプロパティの設定

※新しいプロジェクトファイル（標準EXE）を用意します。

1 ［プロジェクト］メニューの［参照設定］を選び、［Microsoft Excel 9.0 Object Library］を選ぶ

2 フォームの［Caption］プロパティを「為替計算プログラム」に変更する

3 ラベルコントロールを３つ配置し、各コントロールの［Caption］プロパティを変更する

コントロール	［Caption］プロパティ
Label1	現在の為替レート　：１ドル
Label2	¥
Label3	$

Excel97の場合

［Microsoft Excel 8.0 Object Library］を選んでください。

ラベルコントロールを配置する

ラベルコントロールを配置するには、ツールパレットの［Label］ A を使います。

テキストボックスを配置する

テキストボックスコントロールを配置するには、ツールパレットの［TextBox］ abl を使います。

ラベルコントロールを配置する

ラベルコントロールを配置するには、ツールパレットの［Label］ A を使います。

コマンドボタンを配置する

コマンドボタンを配置するには、ツールパレットの［CommandButton］ を使います。

4 テキストボックスコントロールを２つ配置し、各コントロールのプロパティを変更する

コントロール	[Text] プロパティ	[TabIndex] プロパティ
Text1	107	2
Text2	(なし)	0

5 ラベルコントロールを配置し、次のプロパティを変更する

プロパティ	値
BorderStyle	1 ー実線
Caption	(なし)

6 コマンドボタンコントロールを２つ配置し、各コントロールのプロパティを変更する

コントロール	オブジェクト名	Caption	Enabled
Command1	CalcBtn	換算	True
Command2	ExitBtn	終了	True

為替計算プログラム

現在の為替レート ：1ドル　107

¥

$

換算

終了

配置してプロパティを設定したコマンドボタンコントロール

コードの記述

Form1:(General)-(Declarations)

Lesson35 - Form1 (ｺｰﾄﾞ)

(General)　(Declarations)

```
Dim AppXLS As Object
```

Form1:CalcBtn-Click

Lesson35 - Form1 (ｺｰﾄﾞ)

CalcBtn　Click

```
Private Sub CalcBtn_Click()
    Dim Money As Double

    If Text2.Text = "" Then
        Text2.SetFocus
        Exit Sub
    End If

    Money = CLng(Text2.Text) / CLng(Text1.Text)

    Label4.Caption = AppXLS.WorksheetFunction.Dollar(Money)
    Text2.Text = ""
    Text2.SetFocus
End Sub
```

Form1:Form-Load

Lesson35 - Form1 (ｺｰﾄﾞ)

Form　Load

```
Private Sub Form_Load()
    On Error GoTo FAIL
    Set AppXLS = CreateObject("Excel.Application")
```

```
    With Form1
        .Caption = .Caption & "  -  Excelを起動中です"
    End With
    Exit Sub

FAIL:
    MsgBox Error(Err.Number)
End Sub
```

Form1:ExitBtn-Click

Lesson35 - Form1 (コード)

ExitBtn | Click

```
Private Sub ExitBtn_Click()
    Unload Me
    End
End Sub
```

Form1:Form-Unload

Lesson35 - Form1 (コード)

Form | Unload

```
Private Sub Form_Unload(Cancel As Integer)
    AppXLS.Quit
    Set AppXLS = Nothing
End Sub
```

解　説

1　このプログラムは、それほど難しくありません。前回のレッスンで使用したテクニックをそのまま使用して作成します。このプログラムは、入力された「¥」を、ドルに換算して出力します。

①「¥」のテキストボックスコントロールに、日本円を入力します。
②［換算］ボタンをクリックすると、あらかじめ設定されているレートに従って、入力された金額をドルに換算します。
③Excelのワークシート関数「Dollar」を使用して、「$xx.xx」とドルとセントの表記に変換して、ラベルコントロールに出力します。
④フォーム起動時に、CreateObject関数でExcelオブジェクトのインスタンスを作成します。しかし、これまで行ったように、新しいブックを追加する操作はしません。Excelのインスタンスを作成するだけで、ワークシート関数を利用します。

2　では、コードを作成していきましょう。最初に、フォームの宣言セクションに、オブジェクト変数「AppXLS」を宣言します。次に、フォームのLoadイベントプロシージャで、CreateObject関数を使って、Excelのインスタンスを作成します。そして、フォームのキャプションに、Excelを起動している旨の表示を行います。

```
Private Sub Form_Load()
    On Error GoTo FAIL
    Set AppXLS = CreateObject("Excel.Application")

    With Form1
        .Caption = .Caption & "  -  Excelを起動中です"
    End With
    Exit Sub

FAIL:
    MsgBox Error(Err.Number)
End Sub
```

このプロシージャ全体は、前のレッスンと同じコードですが、ただ1つ違うのは、Excelに新しいブックを追加していないことです。今回のプログラムでは、ブックを使わなくても、ワークシート関数を使うことができます。

3　［換算］ボタンで、実際の円－ドル換算の計算を実行します。まず、計算結果を格納する変数「Money」を宣言します。そして、テキストボックスの入力が空なのに［換算］ボタンがクリックされた場合は、プロシージャを終了させる処理を組んでおきます。

```
Private Sub CalcBtn_Click()
    Dim Money As Double

    If Text2.Text = "" Then
        Text2.SetFocus
        Exit Sub
    End If
```

換算計算はとても単純で、入力された金額を為替のレートで割るだけです。このとき、テキストボックスの［Text］プロパティは「文字列」で値を返しますから、CLng関数で長整数型に一度変換してから、割り算を実行します。割り算の結果によっては、小数点が発生しますから、変数「Money」は、倍精度の実数型で宣言しています。

```
Money = CLng(Text2.Text) / CLng(Text1.Text)
```

ワンポイントアドバイス

為替レートの値は、テキストボックスコントロールを使用していますので、常に最新のレートを入力して、式に使用できるようにしています。

4 換算したドルの金額は、Excelのワークシート関数「Dollar」でドル表記に変換します。「Dollar」関数は、引数に金額を設定すると、その金額を必ず「$xx.xx」形式で返してきます。たとえば、引数に「10.2511」という数字が与えられて関数が実行されると、「$10.25」と小数点2桁に丸めて、$記号を付けた「文字列」を返してきます（デフォルトで小数点2桁に丸めます）。

Dollar関数は、桁数の丸め処理とドル表記を一度に行ってくれるので、面倒な式や文字列処理がいりません。ここでは、計算結果をDollar関数でドル表記に変換して、ラベルコントロールに出力しています。

```
Label4.Caption = AppXLS.WorksheetFunction.Dollar(Money)
```

そして、テキストボックスの入力フィールドをクリアし、フォーカスを戻して次の入力に備えます。

```
Text2.Text = ""
Text2.SetFocus
```

COLUMN

イミディエイトウィンドウで戻り値を調べる

次の画面は、このプログラムをステップ実行して、計算結果の入っている変数「Money」と、Dollar関数の戻り値であるラベルコントロールの［Caption］プロパティの値を、イミディエイトウィンドウに表示しています。見てわかるように、Dollar関数の戻り値は、小数点の桁数が2桁に丸められて返されています。

ステップ実行しているところ

丸められた戻り値

ワンポイントアドバイス

「Dollar」関数で、戻り値に小数点の桁数を指定したい場合は、第二引数として桁数を数字で指定します。次のDollar関数は、４桁の小数点を付けて返してきます。

Money = 15.2244137
Dollar(Money,4)
Dollar関数の戻り値は、「$15.2244」になります。

5 最後に、プログラムの終了処理を作成します。フォームのUnloadイベントプロシージャでは、Excelのインスタンスを「Quit」メソッドで終了させています。このプログラムでは、ブックを1つも開いていませんので、直接Excelを終了させるメソッドを使っています。

```
Private Sub ExitBtn_Click()
    Unload Me
    End
End Sub

Private Sub Form_Unload(Cancel As Integer)
    AppXLS.Quit
    Set AppXLS = Nothing
End Sub
```

まとめ

ワークシート関数は、大変種類が豊富ですから、いろいろな使い方ができます。しかし、Excelで使用できるすべての関数が、Visual Basicから利用できるわけではありません。中には、Visual Basicで使えない関数もあるので、「付録　Visual Basicで使用できるワークシート関数の一覧」を参照して、プログラムの機能にあったワークシート関数を使ってください。

練習問題

Q1 為替レートを、リストボックスから選べるようにし、換算のシミュレーションができるように修正してください。

解答は巻末に→

Lesson 36

第10日
5時限目

Excelを
Visual Basic
で操作する

Lesson36.vbp
VB6CD2
Lesson36

フォトアルバムの作成

このレッスンでは、Excelの操作とVisual Basicのコントロールを組み合わせて、フォトアルバムを作成します。Excelのワークシートには、写真のファイル名とカメラ、撮影場所・年月日などが入力されています。そして、写真をJPEGファイルにしたものもあります。このプログラムでは、Excelのワークシートから撮影データを読み込んでフォームに表示するとともに、対応するJPEGファイルをイメージコントロールに表示して、写真とデータを同時に見られるようにしてます。

今回作成するプログラム

	A	B	C	D	E
1	写真名	ファイル名	カメラ	レンズ	撮影日
2	ロビー	airport.jpg	PENTAX Z-50P	28-80mm Power Zoom	1997
3	時計の建物	clock.jpg	Canon A1	135mm	1996
4	馬と子供	hose.jpg	Canon A1	85mm	1996
5	バラープレイボーイ	rose-1.jpg	Canon A1	85mm + クローズアップフィルタ	1995
6	窓と花	window.jpg	PENTAX Z-50P	28-80mm Power Zoom	1997

Excelにはワークシートに撮影データが入力されている

フォルダには、写真をJPEGファイルにして保存してある

プログラムは、ワークシートからデータを読み込み、対応するJPEGファイルを表示する

POINT

このプログラムでは、Excelを起動し、ワークシートから撮影データを取得して、フォームに配置したラベルコントロールに表示します。また、データに対応する写真を読み込み、イメージコントロールに表示します。Excelは、起動しても表示せずに、完全にバックグラウンドのプロセスとして動作させます。そして、Excelはプログラム終了時に、プログラムから操作して終了させます。

コントロールとプロパティの設定

※新しいプロジェクトファイル（標準EXE）を用意します。付録CD-ROMの「Lesson36」フォルダにある次のファイルを、プログラムを作成するフォルダにコピーしてください。
● photoalbum.XLS ● CAMERA.ICO ● airport.jpg ● clock.jpg
● hose.jpg ● rose-1.jpg ● window.jpg

1 ツールパレットに「Microsoft Windows Common Control 6.0」コントロールを追加する

ツールパレットにコンポーネントが追加される

コンポーネントを追加する

コンポーネントを追加する方法については、「Lesson5 ツールバーコントロールの組み込み」を参照してください。

2 フォームを広げ、［Caption］プロパティを「Custom Photo Album」に変更する

Custom Photo Album

［Caption］プロパティを変更して、サイズを広げたフォーム

3 ［Icon］プロパティの［...］ボタンをクリックし、「CAMERA.ICO」を設定する

アイコンの読み込み
ファイルの場所(I): material
264.ico 266.ico 48.ico 50.ico 52.ico 54.ico 56.ico 58.ico Atoms2.ico Button13.ico Buttonlf.ico Buttonrg.ico Calc.ico Camera.ico center.ico check.ico Clipbrd2.ico Clock8.ico Copya.ico Disk_0g.ico Disk_3i.ico Disk_3j.ico Diskcat2.ico Doc05.ico
ファイル名(N): Camera.ico
ファイルの種類(T): アイコン (*.ico;*.cur)
開く(O)　キャンセル　ヘルプ(H)

1 ここをクリック

2 ここをクリック

[...] ボタン

[...] ボタンは、そのプロパティの詳細な指定を行うダイアログボックスが用意されていることを示しています。

コマンドボタンを配置する

コマンドボタンを配置するには、ツールパレットの［CommandButton］を使います。

フレームコントロールを配置する

フレームコントロールを配置するには、ツールパレットの［Frame］を使います。

4 コマンドボタンコントロールを2つ配置し、各コントロールのプロパティを変更する

コントロール	オブジェクト名	［Caption］プロパティ
Command1	OpenBtn	アルバムを開く
Command2	ExitBtn	終了

Custom Photo Album
アルバムを開く
終了

配置してプロパティを設定したコマンドボタンコントロール

5 フレームコントロールを配置し、［Caption］プロパティを「撮影データ」に変更する

Custom Photo Album
アルバムを開く
終了
撮影データ

配置してプロパティを設定したフレームコントロール

6 ツールーパレットの［ListBox］を使って、リストボックスコントロールをフレーム内に配置する

Custom Photo Album
アルバムを開く
終了
撮影データ

1 斜めにドラッグして配置

7 [List] プロパティをクリックしてを文字を消す

プロパティ－List1	
List1 ListBox	
全体	項目別
HelpContextID	0
IMEMode	0－なし
Index	
IntegralHeight	True
ItemData	(ﾘｽﾄ)
Left	240
List	(ﾘｽﾄ)
MouseIcon	
MousePointer	

1 ここの文字を削除

8 ラベルコントロールを配置し、次のプロパティを設定する

プロパティ	値
Alignment	2 －中央揃え
BorderStyle	1 －実線
Caption	(なし)

配置してプロパティを設定したラベルコントロール

9 作成したラベルコントロールを［コピー］と［貼り付け］を使って３つコピーし、サイズを調整する

コピーしてサイズを調整したラベルコントロール

ラベルコントロールを配置する

ラベルコントロールを配置するには、ツールパレットの［Label］A を使います。

ラベルコントロールを配置する

ラベルコントロールを配置するには、ツールパレットの［Label］A を使います。

10 ラベルコントロールを4つ配置し、各コントロールの［Caption］プロパティを設定する

コントロール	［Caption］プロパティ
Label5	カメラ
Label6	レンズ
Label7	撮影日時
Label8	撮影場所

配置してプロパティを設定したラベルコントロール

イメージコントロールを配置する

イメージコントロールを配置するには、ツールパレットの［Image］を使います。

11 イメージコントロールを配置し、［BorderStyle］プロパティを「1－実線」に設定する

配置してプロパティを設定したイメージコントロール

12 ツールーパレットの［StatusBar］を使って、ステータスバーコントロールを配置する

13 ［プロパティページ］の［パネル］タブで［サイズの自動設定］を「1 – sbrSpring」に設定する

14 ［プロジェクト］メニューの［参照設定］を選び、［Microsoft Excel 9.0 Object Library］を選ぶ

［プロパティページ］ダイアログ

［プロパティページ］ダイアログボックスを表示するには、コントロールを選択し、プロパティウィンドウの［(プロパティページ)］をクリックし、右欄の［...］ボタンをクリックします。

Excel97の場合

［Microsoft Excel 8.0 Object Library］を選んでください。

コードの記述

Form1:(General)-(Declarations)

Lesson36 - Form1 (コード)

(General) | (Declarations)

```
Dim MyXLS As Object
Dim AlbumBook As Object
Dim DetectXLS As Boolean
```

Form1:Form-Load

Lesson36 - Form1 (コード)

Form | Load

```
Private Sub Form_Load()
    DetectXLS = False
End Sub
```

Form1:OpenBtn-Click

Lesson36 - Form1 (コード)

OpenBtn | Click

```
Private Sub OpenBtn_Click()
    Dim i As Integer
    Dim XLSfile As String

    i = 2
    XLSfile = App.Path & "¥photoalbum.xls"
    On Error GoTo FAIL
    Set MyXLS = CreateObject("Excel.Application")
    Set AlbumBook = MyXLS.Workbooks.Open(XLSfile)
    Do
        Str1 = AlbumBook.Worksheets("Sheet1").Cells(i, 1)
        List1.AddItem Str1
        i = i + 1
    Loop While Str1 <> ""

    List1.RemoveItem i - 3
    StatusBar1.Panels(1).Text = "Excelファイルを開いています"
    OpenBtn.Enabled = False
    DetectXLS = True
    Exit Sub

FAIL:
    MsgBox Error(Err.Number)
End Sub
```

Form1:List1-Click

Lesson36 - Form1 (コード)

List1 / Click

```
Private Sub List1_Click()
  Dim Pos As Integer
  Dim Fname As String

  Pos = List1.ListIndex + 2
  With AlbumBook.Worksheets("Sheet1")
    Label1(0).Caption = .Cells(Pos, 3)
    Label1(1).Caption = .Cells(Pos, 4)
    Label1(2).Caption = .Cells(Pos, 5)
    Label1(3).Caption = .Cells(Pos, 6)
    Fname = .Cells(Pos, 2)
  End With

  Fname = App.Path & "¥" & Fname
  Image1.Picture = LoadPicture(Fname)
End Sub
```

Form1:Form-Unload

Lesson36 - Form1 (コード)

Form / Unload

```
Private Sub Form_Unload(Cancel As Integer)
    Form1.SetFocus

    On Error Resume Next
    If DetectXLS = True Then
        MsgBox "開いたExcelのBookを閉じます", vbOKOnly, "Excelの終了"
        ActiveWorkbook.Close
        MyXLS.Quit
        Set MyXLS = Nothing
        Set AlbumBook = Nothing
    End If
End Sub
```

Form1:ExitBtn-Click

Lesson36 - Form1 (コード)

ExitBtn / Click

```
Private Sub ExitBtn_Click()
    Unload Me
    End
End Sub
```

解 説

1 このプログラムは、写真の撮影データをExcelのブックから読み込み、写真はイメージファイルから読み込んで、それぞれを1つのフォームに表示するプログラムです。実際のアルバムでは、撮影データは常に追加されていきます。そのためもし、Visual Basic単独でアルバムプログラムを作ったとすると、撮影データを変更するたびにプログラムを作り直さなければなりませんから現実的ではありません。また、撮影データをテキストファイルに保存して、シーケンシャルまたはランダムアクセスで読み込む方法もありますが、データの入出力処理を組み立てる必要があります。

このプログラムは、外部データベースとしてExcelのワークシートを使うことで、データの入力処理を容易にしています。プログラム起動時に、常にセルに入力されている撮影データを読み込みますので、撮影データが増減してもブックを開いた時点の最新のデータを読み込むようになっています。また、データの中に写真のイメージファイル名（パス名）を記述しておき、これを読み込んでイメージコントロールに表示していますから、簡単にデータと写真を一致させて表示できます。

Microsoft Excel - photoalbum.xls

	A	B	C	D	E
1	写真名	ファイル名	カメラ	レンズ	撮影日
2	ロビー	airport.jpg	PENTAX Z-50P	28-80mm Power Zoom	1997年8月
3	時計の建物	clock.jpg	Canon A1	135mm	1996年7月
4	馬と子供	hose.jpg	Canon A1	85mm	1996年8月
5	バラープレイボーイ	rose-1.jpg	Canon A1	85mm + クローズアップフィルタ	1995年6月
6	窓と花	window.jpg	PENTAX Z-50P	28-80mm Power Zoom	1997年8月

撮影データはすべてワークシートで管理する

2 プログラムの動作を説明しておきます。

①フォームにある［アルバムを開く］ボタンをクリックすると、「CreateObject」関数を使って、Excelのインスタンスを作成します。そして、ExcelのOpenメソッドで、プログラムと同じフォルダにあるブック「photoalbum.xls」を開きます。この時点で、Excelを表示にする処理を行っていませんから、プログラムが終了するまでExcelは一切表示されず、バックグラウンドプロセスとして動作します。

②ブックを開いたら、ワークシート「Sheet1」にあるセルを1行目から順番にサーチし、写真の名前をフォームのリストボックスに組み込みます。

Custom Photo Album
アルバムを開く
終了
撮影データ
ロビー
時計の建物
馬と子供
カメラ
レンズ
撮影日時
撮影場所
Excelファイルを開いています

写真の名前をリストボックスに組み込む

③リストボックスへの組み込みが終了すると、［アルバムを開く］ボタンを無効にして、二重にファイルが読み込まれることを防止しておきます。

このボタンを無効にする

④リストボックスで写真の名前がクリックされると、その行番号を使って、セルからカメラやレンズなどの撮影データを読み込み、ラベルコントロールに表示します。

写真の名前をクリックすると…

撮影データがラベルコントロールに表示される

⑤更に、同じ行に入力されているイメージファイル名を取得し、イメージコントロールに表示します。

続けてイメージコントロールにイメージが表示される

ワンポイントアドバイス

Excelは、複数のインスタンスを作成できるように作られていますので、このプログラムでExcelを使用していても、別のExcelを起動できます。ただし、同じファイルを開こうとすると、ファイルが使用中である旨のメッセージが表示されます。

使用中のファイル

photoalbum.xls は編集のためロックされています。

使用者は '瀬戸 遥' です。

ドキュメントのコピーを読み取り専用で開き、ほかの人がファイルの使用を終了したときに通知を受け取るには、[通知] をクリックします。

[読み取り専用(R)] [通知(N)] [キャンセル]

3 では、順番にプログラムを作成していきましょう。フォームの宣言セクションに、2つのオブジェクト変数をモジュールレベルの変数として宣言します。

```
Dim MyXLS As Object
Dim AlbumBook As Object
Dim DetectXLS As Boolean
```

「MyXLS」は、Excelのオブジェクトを格納します。「AlbumBook」は、開いたブックのオブジェクト名を格納します。このプログラムでは、ワークシートやセルへのアクセスは、ブックのオブジェクト変数を使って行います。「DetectXLS」はこのプログラムがExcelを使用したかどうかを確かめるために使用する変数です。

4 プログラム起動時にはまだExcelを開いていませんから、「DetectXLS」変数に「False」を設定しておきます。これは後でフォームをアンロードするときにフラグ変数として使用します。

```
Private Sub Form_Load()
    DetectXLS = False
End Sub
```

5 次に、[アルバムを開く] ボタンの処理を組み立てます。このボタンのClickイベントプロシージャでは、撮影データを格納しているブック「photoalbum.xls」を開き、写真名をリストボックスに組み込みます。

まず、変数を2つ用意します。「i」は、ワークシートのセルを参照する際の行方向のセル番地を格納します。「XLSfile」は、開くブックのファイル名をフルパスで格納する変数です。

```
Private Sub OpenBtn_Click()
    Dim i As Integer
    Dim XLSfile As String
```

変数を宣言したら、変数「i」を初期化しておきます。ブック「photoalbum.xls」は、1行目が項目見出しになっているため、2行目から実際のデータが入力されています。この行番号を、変数「i」に設定しています。

```
i = 2
```

	A	B	C	D	E	F
1	写真名	ファイル名	カメラ	レンズ	撮影日	撮影場
2	ロビー	airport.jpg	PENTAX Z-50P	28-80mm Power Zoom	1997年8月	羽田空
3	時計の建物	clock.jpg	Canon A1	135mm	1996年7月	清里
4	馬と子供	hose.jpg	Canon A1	85mm	1996年8月	八ヶ岳
5	バラープレイボーイ	rose-1.jpg	Canon A1	85mm + クローズアップフィルタ	1995年6月	自宅
6	窓と花	window.jpg	PENTAX Z-50P	28-80mm Power Zoom	1997年8月	北海道
7						

実際のデータは2行目から入力されている

そして、変数「XLSfile」に、ブック「photoalbum.xls」をフルパスで代入します。「photoalbum.xls」は、プログラムと同じフォルダにコピーしていますから、Appオブジェクトの［Path］プロパティの値と結合して格納します。

```
XLSfile = App.Path & "¥photoalbum.xls"
```

6 開くファイル名が設定できたら、「CreateObject」関数でExcelのインスタンスを作成します。そして、戻り値のオブジェクト名を、オブジェクト変数「MyXLS」に格納します。続いて、ExcelのOpenメソッドでブック「photoalbum.xls」を開きます。Openメソッドは、実行するとブックのオブジェクト名を返してきますから、これをもう1つのオブジェクト変数「AlbumBook」に格納します。

```
On Error GoTo FAIL
Set MyXLS = CreateObject("Excel.Application")
Set AlbumBook = MyXLS.Workbooks.Open(XLSfile)
```

なお、「CreateObject」関数がオブジェクトのインスタンスを作成できなかった場合に備えて、「On Error GoTo」ステートメントでエラー処理を作成しておきます。また、ここではブックを開いても［Visible］プロパティで表示する操作をしていません。表示していなくても、ワークシートやセルへのアクセスは可能で、バックグラウンドでExcelを動作させた処理が可能になります。この処理は、ユーザーに不要な操作をさせない、という点で便利です。

7 ブックを開いたら、ワークシート「Sheet1」のセルを、2行目から順番にサーチして、「写真名」をリストボックスのリスト項目に追加していきます。

まず、セルへのアクセスですが、［Range］プロパティを使わずに「Cells」プロパティでアクセスします。［Range］プロパティが「A1」形式でセル番地を指定するのに対し、［Cells］プロパティは「1、1」形式でセル番地を指定できます。たとえば、セル「A1」にアクセスするには、「Cells(1,1)」と記述します。引数の最初の数字が行番号で、次の数字が列番号になります。

```
Cells(Row,Column)
```

セル番地に数字が使えるということは、変数が使えるということであり、変数が使えるということはループ処理などの自動化ができる、ということになります。

そこで、Do...Loopステートメントを使用して、セルを2行目から1行ずつ順番にアクセスし、セルの値を変数「Str1」に格納します。「写真名」は1列目（A列）に入力されていますから、「Cells(i, 1)」という式を組み立て、変数「i」の値を変化させていけば、自動的にセル番地を1行ずつずらすことができます。

この仕組みを利用して、Do...Loopステートメントの最初に、まず2行目の1列にあるセルの値を変数「Str1」に格納します。変数には、最初の写真名「ロビー」が格納されます。そして、その写真名をフォームに配置したリストボックスの項目として、AddItemメソッドを使って組み込みます。そして、変数「i」の値を1つ増やしておきます。これで、次にループがまわると、次の行のセルにアクセスします。これを、セルの値が空白の行に出会うまで、ループを繰り返します。

Microsoft Excel - photoalbum.xls

	A	B	C	D	E	F
1	写真名	ファイル名	カメラ	レンズ	撮影日	撮影場
2	ロビー	airport.jpg	PENTAX Z-50P	28-80mm Power Zoom	1997年8月	羽田空
3	時計の建物	clock.jpg	Canon A1	135mm	1996年7月	清里
4	馬と子供	hose.jpg	Canon A1	85mm	1996年8月	八ヶ岳
5	バラープレイボーイ	rose-1.jpg	Canon A1	85mm + クローズアップフィルタ	1995年6月	自宅
6	窓と花	window.jpg	PENTAX Z-50P	28-80mm Power Zoom	1997年8月	北海道
7						

「写真名」は2行目のセル「A2」から格納されている

Custom Photo Album
アルバムを開く
終了
撮影データ
ロビー
時計の建物
馬と子供
カメラ
レンズ
撮影日時
撮影場所
Excelファイルを開いています

写真名が順番にリストボックスに組み込まれる

```
Do
    Str1 = AlbumBook.Worksheets("Sheet1").Cells(i, 1)
    List1.AddItem Str1
    i = i + 1
Loop While Str1 <> ""
```

8　値が空白のセルに出会うと、そこでループは終了します。しかし、リストボックスへのAddItemメソッドによる項目の組み込みは、この空白までも行われてしまいます。そこで、この空白のリスト項目を削除しておきます。

リストボックスコントロールの「RemoveItem」メソッドは、リストボックスからリスト項目を削除します。引数に削除するリスト項目のインデックス番号を指定します。リストボックスに組み込んだ項目の数は、変数「i」の最終値ですが、「i」は最初に「2」で初期化していることと、リスト項目の先頭のインデックス番号が「0」から始まっていることを組み合わせると、最後に追加した空白のリスト項目のインデックス番号は、「i - 3」になります。これを、「RemoveItem」メソッドの引数に指定して実行すると、最後の空白のリスト項目を削除できます。

```
List1.RemoveItem i - 3
```

9　リストボックスのリスト項目が追加できたら、ステータスバーコントロールにブックを開いていることを表示します。Excelは、すべて非表示の状態で起動しています。ユーザーの目にはExcelは見えませんから、ブックを開いていることを表示するようにします。そして、［アルバムを開く］ボタンの［Enabled］プロパティを「False」に設定し、同じファイルが2回開かれないようにしておきます。また、ここでExcelを開いたので「DetectXLS」には「True」を設定しておきます。

```
StatusBar1.Panels(1).Text = "Excelファイルを開いています"
OpenBtn.Enabled = False
DetectXLS = True
Exit Sub
```

プロシージャの最後に、エラー処理でジャンプさせたラベル「FAIL」の処理を組みます。これは、Error関数とErrオブジェクトの［Number］プロパティを組み合わせて、発生したエラーのメッセージを表示させてプロシージャを終了させます。

```
FAIL:
    MsgBox Error(Err.Number)
End Sub
```

10 今度は、リストボックスの項目がクリックされたときの処理を作成します。ここでは、写真名がクリックされれば、そのセルと同じ行から撮影データを読み出し、ラベルコントロールに出力します。また、同じ行に入力されている写真のイメージファイル名を読み出し、イメージコントロールで表示します。リストボックスの項目が、ユーザーによってクリックされると、Clickイベントが発生します。そこで、このイベントプロシージャで、これらの処理を行います。

まず、2つの変数を宣言します。「Pos」は、セルの行番号を格納します。「Fname」は、イメージコントロールに設定する写真のファイル名を格納するのに使用します。

```
Private Sub List1_Click()
    Dim Pos As Integer
    Dim Fname As String
```

変数を宣言したら、最初にリストボックスで選択されている項目名、つまり写真名を取得します。リストボックスコントロールの［ListIndex］プロパティには、選択されたリスト項目のインデックス番号が格納されています。この番号を使って、セルの行番号を導き出します。先ほど行ったリスト項目の設定と逆の処理です。

リスト項目の先頭がクリックされると、［ListIndex］プロパティには「0」が格納されます。この項目は、ワークシートの2行目のセルから転送していますから、セルの行番号は「ListIndex + 2」になります。

```
Pos = List1.ListIndex + 2
```

このようにして作成した行番号を使って、［Cells］プロパティでセルにアクセスします。撮影データは、写真名の行の3、4、5、6列に格納されていますから、順番に取り出してラベルコントロールの［Caption］プロパティに代入します。また、写真のイメージファイル名は、同じ行の2列目に入力されていますから、これを変数「Fname」に代入しておきます。

```
With AlbumBook.Worksheets("Sheet1")
    Label1(0).Caption = .Cells(Pos, 3)
    Label1(1).Caption = .Cells(Pos, 4)
    Label1(2).Caption = .Cells(Pos, 5)
    Label1(3).Caption = .Cells(Pos, 6)
    Fname = .Cells(Pos, 2)
End With
```

			③	④	⑤	⑥	
	A	B	C	D	E	F	
1	写真名	ファイル名	カメラ	レンズ	撮影日	撮影場所	コ
2	ロビー	airport.jpg	PENTAX Z-50P	28-80mm Power Zoom	1997年8月	羽田空港	旅
3	時計の建物	clock.jpg	Canon A1	135mm	1996年7月	清里	清
4	馬と子供	hose.jpg	Canon A1	85mm	1996年8月	八ヶ岳高原	ハ
5	バラープレイボーイ	rose-1.jpg	Canon A1	85mm + クローズアップフィルタ	1995年6月	自宅	自
6	窓と花	window.jpg	PENTAX Z-50P	28-80mm Power Zoom	1997年8月	北海道	北

イメージファイル名は、プログラムと同じフォルダに格納してありますので、Appオブジェクトの［Path］プロパティと結合して、フルパス名に変換します。そして、このファイル名を、イメージコントロールの［Picture］プロパティに、LoadPicture関数で設定します。

```
Fname = App.Path & "¥" & Fname
Image1.Picture = LoadPicture(Fname)
```

このようにして、リストボックスの写真名をクリックすれば、ワークシートの同じ行にある撮影データとイメージファイル名を、各コントロールに設定して表示できるようになります。

リスト項目をクリックすれば、撮影データとイメージファイルが表示される

11　最後に、プログラムの終了処理を組み立てます。まず、［終了］ボタンでは、おなじみのプログラムの終了処理を記述します。

```
Private Sub ExitBtn_Click()
    Unload Me
    End
End Sub
```

12　しかし、今回のプログラムでは、これだけでは終了処理になりません。まず、ユーザーがExcelを起動させたのかどうか調べる必要があります。［アルバムを開く］ボタンをクリックせずに［終了］ボタンを押す可能性もあります。

また、起動した場合は、Excelのインスタンスオブジェクトは、まだ起動したままになっています。ユーザーにExcelを表示していないので、ユーザーが自分でExcelを終了させることができません。プログラムで責任を持ってExcelを終了させる必要があります。

注意しなければならないのは、この処理を［終了］ボタンに組み込んでしまうと、フォームの［X］ボタンをクリックしてプログラムを終了したときに、Excelの終了処理が行われなくなってしまうことです。そこで、この処理をフォームのUnloadイベントプロシージャに組み込みます。プログラムが終了する際に、フォームがメモリから削除される前に、フォームには「Unload」イベントが発生します。このイベントプロシージャに、Excelを終了させる処理を組み込んでおけば、プログラム終了直前に、この処理が実行されます。

まず、いったんフォームにフォーカスを戻します。

```
Private Sub Form_Unload(Cancel As Integer)
    Form1.SetFocus
```

そして、変数「DetectXLS」の値を調べて「True」(Excelが起動された）の状態なら、処理を続行します。もし「False」の状態なら、そのままIf文を抜け出てなにもせずに終了します。また、エラーが起こったときのためにIf文の前に「On Error Resume Next」文を追加しておきます。これでエラーが起こっても、エラーはスキップされ次の文へ処理が移ります。

```
On Error Resume Next
If DetectXLS = True Then
```

処理を知らせるメッセージボックスを表示します。処理の分岐がないのでボタンは［OK］ボタンだけにします。

```
MsgBox "開いたExcelのBookを閉じます", vbOKOnly, "Excelの終了"
```

Excelを終了させるには、開いたブックを閉じさせ、Quitメソッドを実行します。これで、ブックが閉じてExcelが終了します。

```
ActiveWorkbook.Close
MyXLS.Quit
```

最後に、作成したオブジェクト変数に「Nothing」を代入して、解放します。

```
    Set MyXLS = Nothing
    Set AlbumBook = Nothing
End If
```

まとめ

Visual Basic のプログラムと Excel をうまく組み合わせて使用する一例を紹介しました。今回のように、撮影データとイメージファイルがバラバラに存在するような場合は、Visual Basic のプログラムでこれらを統合するようなアプリケーションを開発すれば、効果的でしょう。Excel を使うことができれば、すでにあるリソースを有効に活用できますから、開発の負担も軽くなります。

まだまだ、アイデア次第でいろいろな活用法があると思います。リストボックスのインデックス番号とセル番地の数字の不一致などは、式を組み立てれば簡単に解消できます。

練習問題

Q1　リストボックスの項目をクリックすると、写真名と同じ行にあるコメントをステータスバーコントロールに表示するように修正してください。

写真名をクリックすると、ステータスバーコントロールにコメントが表示される

解答は巻末に→

ワンポイントアドバイス

すでに説明したように、このプログラムでは Excel をバックグラウンドプロセスとして起動しています。スクリーンには表示されませんが、Ctrl+Alt+Delete キーを押すことで起動中の Excel を確認することができます。

Excel が起動している

10日でおぼえる

Visual Basic 6.0 実践教室

Let's VBSchool!!

付録
各種リファレンス

Visual Basic プログラミングに役立つ各種リファレンスを掲載しました。

「10日でおぼえるVisual Basic6.0入門教室」の方には便利な用語集や関数一覧、エラーコード集などがありますので、そちらもあわせて参照してください。

巻末に各レッスンの練習問題の解答が付いています。

●付録

- ❏ Visual Basic で Access2000 のデータベースファイルを操作する方法について
- ❏ SystemParametorsInfo について
- ❏ Visual Basic で使用できるワークシート関数一覧
- ❏ 練習問題の解答

Visual Basic で Access2000 のデータベースファイルを操作する方法について

ここでは、Microsoft Access2000 で作成されたデータベースファイルを、Visual Basic で操作する方法について説明します。

Microsoft Access2000で作成されたデータベースファイルを、Visual Basicで操作する場合、Visual Basicに用意されている「Dataコントロール」では、Access2000のファイル形式を認識できず、データベースファイルの操作が一切できません。

Access2000は、「Microsoft Jet4.0」という新しいデータベースエンジンで動作するアプリケーションであり、これまでのAccess97からファイルフォーマットも変更されています。

一方、Visual Basicに付属する「Dataコントロール」は、内部でDAO 3.5 Object Libraryを使用しており、「Microsoft Jet 4.0」形式のファイルは扱えません。

Microsoft Visual Basic
データベースの形式 'F:¥_GENKOU¥VB60実践¥ADO¥club.mdb' を認識できません。
OK　ヘルプ(H)

Access2000で作成したデータベースファイルを、「Data」コントロールで操作すると表示されるメッセージ

その場合は、「DAO 3.6 Object Library」、バージョン4.0以降の 「ODBC Desktop Driver」、または「Microsoft Jet 4.0 OLE DB Provider」を使用する必要があります。

Excel2000などOffice2000のVBAは、あらかじめ「DAO 3.6 Object Library」を持っており、Visual Basic 6.0も「Visual Studio 6.0 Service Pack 3」をインストールすれば、「DAO 3.6 Object Library」と「Microsoft Jet 4.0 OLE DB Provider」を使用することができます。

また、Visual Basic 6.0に付属する「Microsoft ADO Data Control 6.0 (OLEDB)」を使用すると、この「Microsoft Jet 4.0 OLE DB Provider」を使用して、Access2000のデータベースファイルを操作することが可能になります。ただし、このコントロールは、「データコントロール」のように、「DatabaseName」プロパティでデータベースファイルを指定すれば簡単に操作できる、というわけにはいかず、少し特別な操作が必要になります。

ここでは、「Microsoft ADO Data Control 6.0 (OLEDB)」を使用して、Access2000で作成したデータベースファイルを操作する方法を紹介します。

ID	会員No	氏名	性別	入会年月日	コース名	住所1
1	ADS2113	島村 由美子	女	98/08/07	フルタイム	横浜市戸塚区上菅田1-8-X
2	SFC0012	吉田 昭夫	男	98/03/16	ナイトタイム	横浜市西区西幸町3-32－X
3	KKH2121	泉 陽子	女	99/01/23	デイタイム	藤沢市片瀬2-13－X
4	ALS3522	浅田 智美	女	98/10/18	ナイトタイム	川崎市高津区有馬5-11-X
5	HJJ8819	神山 ゆり	女	98/06/13	ホリディ	東京都大田区東糀谷3-1-X
6	LDW4433	加藤 静雄	男	98/08/11	デイタイム	川崎市川崎区銀柳2-19-X
7	MSF2911	斎藤 仁	男	99/03/03	ナイトタイム	横浜市神奈川区東神奈川9-3-X
8	CBS5401	真鍋 香織	女	98/11/10	フルタイム	東京都品川区品川4-21-X
9	GHD3900	鶴田 美緒	女	98/05/22	デイタイム	横浜市鶴見区花月6-1-X
10	JDB4744	長谷川 隆	男	98/07/18	ナイトタイム	川崎市宮前区寺前2-2-X
11	QWE2022	相川 麻美	女	98/09/30	ナイトタイム	横浜市中区不老町7-1-X
12	NCA0038	安達 園子	女	98/04/04	ホリディ	東京都大田区蒲田4-2-X
13	VWZ2109	山本 静香	女	98/08/12	ナイトタイム	横浜市保土ヶ谷区藤塚町111-X
14	TLU0921	藤森 徹	男	99/03/10	ホリディ	東京都世田谷区深沢4-444-X
15	YVQ3388	西尾 由香	女	98/10/28	ナイトタイム	川崎市川崎区本町13-X
16	MZK0129	浜野 章子	女	99/01/20	フルタイム	川崎市中原区井田21-555－X
(オートナンバー)						

操作対象の「Club.mdb」ファイル。架空のスポーツクラブの会員名簿が格納されている

ワンポイントアドバイス

付録ディスクに収録してある「Club.mdb」ファイルを、使用するコンピュータにコピーし、ファイルのプロパティで「読み取り専用」の属性を外してください。

使用するデータベースファイルが「読み取り専用」のままだと、データベースの接続に失敗する場合があります。

この、「Club.mdb」ファイルは、Microsoft Access2000で作成・保存されています。

ファイルの「読み取り専用」属性を外す

1 新しいプロジェクトファイル（標準EXE）を用意します。そして、ツールボックスの上でマウスの右ボタンを押し、ショートカットメニューから［コンポーネント（O）...］を選びます。

2 ［Microsoft ADO Data Control 6.0(OLEDB)］（またはServicePack3をインストールしている場合は［Microsoft ADO Data Control 6.0（SP3）（OLEDB）］）を選んでチェックボックスをオンにし、［OK］ボタンをクリックします。

3 ツールボックスに「Adodc」というアイコンが表示されます。これが、「ADO」コントロールです。

「ADO」コントロール

Adodc

4 このアイコンをクリックし、フォームにドラッグして配置します。
見た目は、従来の「Data」コントロールとそっくりですが、オブジェクト名が「Adodc」になっているので、その違いがわかります。

Form1

Adodc1

1 ドラッグして配置

5 では、このコントロールに、Microsoft Access2000で作成された「Club.mdb」ファイルを連結させます。プロパティウィンドウにある［プロパティページ］をクリックし、［...］ボタンをクリックします。

プロパティ - Adodc1

Adodc1 Adodc

全体 | 項目別

(オブジェクト名)	Adodc1
(バージョン情報)	
(プロパティページ)	...
Align	0 - vbAlignNone
Appearance	1 - ad3DBevel
BackColor	&H80000005&
BOFAction	0 - adDoMoveFirst
CacheSize	50
Caption	Adodc1
CommandTimeout	30
CommandType	8 - adCmdUnknown

1 ここをクリック

2 ここをクリック

6 ［全般］タブをクリックして、［接続文字列を使用する］をクリックし、［作成］ボタンをクリックします。

7 「データリンクプロパティ」ウィンドウが開きます。［プロバイダ］タブをクリックし、［Microsoft Jet 4.0 OLE DB Provider］をクリックします。そして、［次へ］ボタンをクリックします。

8 ［接続］ページが表示されるので、「1.データベース名を選択または入力します」にある［...］ボタンをクリックします。

9　［Accessデータベースの選択］ダイアログボックスが表示されるので、ここで使用する「Club.mdb」ファイルを選び、［開く］ボタンをクリックします。

1 ここをクリック
2 ここをクリック

10　［ユーザー名］の「Admin」を削除します。そして、［接続のテスト］ボタンをクリックします。

1 ここの文字を削除し...
2 ここをクリック

ワンポイントアドバイス

この「Club.mdb」ファイルは、アクセスするのに特にユーザー名やパスワードを必要としていません。これらが設定されているデータベースファイルに接続する場合は、このページの「ユーザー名」と「パスワード」を設定してください。

11 無事、「Club.mdb」ファイルに接続できれば、その旨のメッセージが表示されます。

［OK］ボタンをクリックしてメッセージを閉じたら、「データリンクプロパティ」ウィンドウの［OK］ボタンをクリックして、ウィンドウを閉じます。

12 プロパティページに戻り、［レコードソース］タブをクリックします。［コマンドタイプ］リストボックスから「2 − adCmdTable」を選びます。

13 自動的に［テーブル名またはストアドプロシージャ名］のリストボックスが有効になりますので、リストから「Club.mdb」ファイルに格納されているテーブル「会員名簿」を選びます。

プロパティページ
全般 | 認証 | レコードソース | 色 | フォント
レコードソース
コマンドタイプ
2 - adCmdTable
テーブル名またはストアドプロシージャ名
会員名簿
会員名簿
コマンドテキスト (SQL)
OK | キャンセル | 適用(A) | ヘルプ

1 ここをクリック

14 これで、ADOコントロールの、「Club.mdb」ファイルに接続するためのプロパティ設定は完了です。［OK］ボタンをクリックして、プロパティページを閉じます。

プロパティページ
全般 | 認証 | レコードソース | 色 | フォント
レコードソース
コマンドタイプ
2 - adCmdTable
テーブル名またはストアドプロシージャ名
会員名簿
コマンドテキスト (SQL)
OK | キャンセル | 適用(A) | ヘルプ

1 ここをクリック

15 Visual Basicに戻り、フォームにラベルコントロールを1つ配置します。そして、プロパティウィンドウでラベルコントロールの「DataSource」プロパティを選び、リストからADOコントロール「Adodc1」を選びます。

これで、ADOコントロールとラベルコントロールが、連結コントロールとして動作するようになります。

Form1
Adodc1
Label1

1 ドラッグして配置

プロパティ - Label1
Label1 Label
全体 | 項目別

DataField	
DataFormat	
DataMember	
DataSource	Adodc1
DragIcon	Adodc1
DragMode	0 - 手動
Enabled	True
Font	MS Pゴシック
ForeColor	&H80000012&
Height	375

1 ここをクリック
2 ここをクリック
3 ここをクリック

16 「DataField」プロパティを選び、リストから「氏名」を選びます。これで、ADOコントロールで選んだレコード位置の、フィールド「氏名」のデータがラベルコントロールに表示されます。

プロパティ - Label1
Label1 Label
全体 | 項目別

AutoSize	False
BackColor	&H8000000F&
BackStyle	1 - 不透明
BorderStyle	0 - なし
Caption	Label1
DataField	氏名
DataFormat	ID
DataMember	会員No
DataSource	氏名
DragIcon	性別
DragMode	入会年月日
	コース名
	住所1

1 ここをクリック
2 ここをクリック
3 ここをクリック

17 プログラムを実行します。今度は、Dataコントロールのようなエラーは表示されずに、きちんとプログラムが動作します。

ADOコントロールのボタンをクリックすれば、データベースに格納されているフィールド「氏名」のデータが、ラベルコントロールに表示されます。

Form1
Adodc1
真鍋 香織

まとめ

このように、Access2000 で作成したデータベースファイルを操作する場合は、Data コントロールではなく、ADO コントロールを使用します。

ADO コントロールではなく DAO を使用するときは、必ず「DAO 3.60」を使用してください。その場合は、プロジェクトの「参照設定」で、「DAO 3.6 Object Library」を指定しないと、Visual Basic はコンパイル時に参照不可でエラーになってしまうので注意してください。

また、Visual Basic 5.0 以前のバージョンでは、ADO も DAO3.60 も使用できないので、Access2000 のデータベースファイルを操作することができません。データベースファイルを旧バージョンの保存形式にして使用するか、バージョン 4.0 以降の ODBC Desktop Driver を使用して、ODBC 経由で接続して使用してください。

参照設定 - Project1

参照可能なライブラリ ファイル(A):

- Microsoft ActiveX Data Objects Recordset 2.0 L
- Microsoft ActiveX Plugin
- Microsoft Add-In Designer
- Microsoft ADO Ext. 2.1 for DDL and Security
- Microsoft Agent Control 2.0
- Microsoft Agent Server 2.0
- Microsoft Agent Server Extensions 2.0
- Microsoft DAO 2.5/3.51 Compatibility Library
- Microsoft DAO 3.51 Object Library
- ✓ Microsoft DAO 3.6 Object Library
- Microsoft Data Adapter Library
- Microsoft Data Binding Collection
- Microsoft Data Environment 1.0
- Microsoft Data Environment Extensibility Object

OK / キャンセル / 参照(B)... / ヘルプ(H) / 優先順位

Microsoft DAO 3.6 Object Library

場所: I¥PROGRAM FILES¥COMMON FILES¥MICROSOFT SHARED¥DAO¥

言語: 標準

DAO 3.60への参照設定

SystemParametersInfo について

SystemParametersInfo()は、システム全体に関係するパラメータを問い合わせたり設定したりするものです。ここでは参考までに簡単な説明だけを行っています。詳細については小社刊『Windows 95 APIバイブル 1』もしくはマイクロソフト社より開発者向けに提供されている『Win32SDK』のドキュメントを参照してください。

Visual Basicで使用できる書式ではこうなっています。

```
Declare Function SystemParametersInfo Lib "user32" _
    Alias "SystemParametersInfoA" _
   (ByVal uAction As Long, _
    ByVal uParam As Long, _
    ByVal lpvParam As Any, _
    ByVal fuWinIni As Long) As Long
```

【SystemParametersInfo()のアクション(uAction)】

値	説明
SPI_GETACCESSTIMEOUT	アクセシビリティ機能のタイムアウトの時間を記述したACCESSTIMEOUT構造体を取得します。
SPI_GETANIMATION	現在のユーザーのアニメーション効果を記述したANIMATIONINFO構造体を取得します。
SPI_GETBEEP	警告のビープ音がオンになっているかどうか調べます。
SPI_GETBORDER	サイズ変更が可能な境界の幅と高さを変更する境界サイズ乗数を取得します。
SPI_GETDRAGFULLWINDOWS	フルウィンドウドラッグが有効になっているかどうかを調べます。
SPI_GETFASTTASKSWITCH	高速タスク切り替えが有効になっているかどうかを調べます。
SPI_GETFILTERKEYS	フィルタキーアクセシビリティ機能を記述したFILTERKEYS構造体を取得します。
SPI_GETGRIDGRANULARITY	Windowsデスクトップ上に配置される項目間の距離を取得します。
SPI_GETHIGHCONTRAST	視覚障害のあるユーザーのためのハイコントラストアクセシビリティ機能を記述したHIGHCONTRAST構造体を取得します。
SPI_GETICONMETRICS	アイコンのメトリックを記述したICONMETRICS構造体を取得します。

値	説明
SPI_GETICONTITLELOGFONT	アイコンのタイトルテキストの表示に使われるフォントを記述したLOGFONT構造体を取得します。
SPI_GETICONTITLEWRAP	アイコンタイトルの自動改行がオンになっているかどうかを調べます。
SPI_GETKEYBOARDDELAY	キーボードのキーリピートディレイ設定を取得します。
SPI_GETKEYBOARDPREF	マウスよりはキーボードの使用を好むユーザーが、アプリケーションで非表示になっているキーボードインターフェイスの表示を希望しているかどうかを調べます。
SPI_GETKEYBOARDSPEED	キーボードのキーリピート速度設定を取得します。
SPI_GETLOWPOWERACTIVE	低電力スクリーンセービングフェーズがアクティブになっているかどうかを調べます。
SPI_GETLOWPOWERTIMEOUT	低電力スクリーンセービングフェーズのタイムアウト値を取得します。
SPI_GETMENUDROPALIGNMENT	ポップアップメニューが対応するメニューバー項目に左揃えされるか右揃えされるかを調べます。
SPI_GETMINIMIZEDMETRICS	最小化されたウィンドウのメトリックを記述したMINIMIZEDMETRICS構造体を取得します。
SPI_GETMOUSE	マウスのスレッショルドと速度を取得します。
SPI_GETMOUSEKEYS	マウスキーアクセシビリティ機能についての情報を取得します。
SPI_GETNONCLIENTMETRICS	NONCLIENTMETRICS構造体内の、最小化されていないウィンドウの非クライアント領域のメトリックを取得します。
SPI_GETPOWEROFFACTIVE	スクリーンセービングの電源切断フェーズが有効になっているかどうかを調べます。
SPI_GETPOWEROFFTIMEOUT	電源切断スクリーンセービングフェーズのタイムアウト値を取得します。
SPI_GETSCREENREADER	テキストデータを点字式ディスプレイや音声シンセサイザにリダイレクトするスクリーンレビューアユーティリティがあるかどうかを調べます。
SPI_GETSCREENSAVEACTIVE	スクリーンセーバーが有効になっているかどうかを調べます。
SPI_GETSCREENSAVETIMEOUT	スクリーンセービングのタイムアウト値を取得します。
SPI_GETSERIALKEYS	シリアルキーアクセシビリティ機能を記述したSERIALKEYS構造体を取得します。
SPI_GETSHOWSOUNDS	ユーザーがサウンドの視覚表現を希望しているかどうかを調べます。

値	説明
SPI_GETSOUNDSENTRY	サウンドセントリアクセシビリティ機能のパラメータを記述したSOUNDSENTRY構造体を取得します。
SPI_GETSTICKYKEYS	スティッキキーアクセシビリティ機能のパラメータを記述したSTICKYKEYS構造体を取得します。
SPI_GETTOGGLEKEYS	トグルキーアクセシビリティ機能のパラメータを記述したTOGGLEKEYS構造体を取得します。
SPI_GETWORKAREA	デスクトップの作業領域のサイズを取得します。
SPI_ICONHORIZONTALSPACING	アイコンの水平方向の間隔を変更します。
SPI_ICONVERTICALSPACING	アイコンの垂直方向の間隔を変更します。
SPI_LANGDRIVER	言語ドライバの名前を取得します。
SPI_SETACCESSTIMEOUT	アクセシビリティ機能のタイムアウト時間を記述したACCESSTIMEOUT構造体を設定します。
SPI_SETANIMATION	現在のユーザーのアニメーション効果を記述したANIMATIONINFO構造体を設定します。
SPI_SETBEEP	警告のビープ音をオン／オフにします。
SPI_SETBORDER	境界サイズ乗数を設定します。
SPI_SETDESKPATTERN	デスクトップの背景パターンをWIN.INIファイルのpatternキーの現在値に設定します。
SPI_SETDESKWALLPAPER	デスクトップの壁紙を設定します。
SPI_SETDOUBLECLICKTIME	1回のダブルクリックとして解釈される2回のクリックの間の時間間隔を設定します。
SPI_SETDOUBLECLKWIDTH	ダブルクリックとして解釈される2回目のクリックの水平距離を設定します。
SPI_SETDOUBLECLKHEIGHT	ダブルクリックとして解釈される2回目のクリックの垂直距離を設定します。
SPI_SETDRAGFULLWINDOWS	フルウィンドウドラッグをオン／オフにします。
SPI_SETFASTTASKSWITCH	高速タスク切り替えをオン／オフにします。
SPI_SETFILTERKEYS	フィルタキーアクセシビリティオプションデータを設定します。
SPI_SETGRIDGRANULARITY	配置する項目間の距離を設定します。
SPI_SETHIGHCONTRAST	ハイコントラストアクセシビリティオプションのパラメータを設定します。
SPI_SETICONMETRICS	アイコンのメトリックを設定します。
SPI_SETICONTITLELOGFONT	アイコンタイトルのテキストのフォントを設定します。
SPI_SETICONTITLEWRAP	アイコンタイトルの自動改行をオン／オフにします。
SPI_SETKEYBOARDDELAY	キーリピートディレイを設定します。

値	説明
SPI_SETKEYBOARDPREF	マウスの代わりに、隠されているキーボードインターフェイスを表示するようにプリファレンスを設定します。
SPI_SETKEYBOARDSPEED	キーボードのキーリピート速度を設定します。
SPI_SETLOWPOWERACTIVE	低電力スクリーンセービングフェーズを設定します。
SPI_SETLOWPOWERTIMEOUT	低電力スクリーンセービングフェーズのタイムアウト値を設定します。
SPI_SETMENUDROPALIGNMENT	メニュードロップアラインメントを設定します。
SPI_SETMINIMIZEDMETRICS	最小化されたウィンドウのメトリックを設定します。
SPI_SETMOUSE	マウスのパラメータを設定します。
SPI_SETMOUSEBUTTONSWAP	マウスの左ボタンと右ボタンの意味を切り替えます。
SPI_SETMOUSEKEYS	マウスキーアクセシビリティオプションのデータを設定します。
SPI_SETNONCLIENTMETRICS	最小化されていないウィンドウの非クライアント領域のメトリックを設定します。
SPI_SETPENWINDOWS	ペンウィンドウをロードするかアンロードするかを指定します。
SPI_SETPOWEROFFACTIVE	スクリーンセービングの電源切断フェーズを設定します。
SPI_SETPOWEROFFTIMEOUT	電源切断スクリーンセービングフェーズのタイムアウト値を設定します。
SPI_SETSCREENREADER	uParam パラメータが TRUE の場合、スクリーンレビューアユーティリティが現在実行されていることを表します。
SPI_SETSCREENSAVEACTIVE	スクリーンセービングを有効または無効にします。
SPI_SETSCREENSAVETIMEOUT	通常のスクリーンセーブのタイムアウト値を設定します。
SPI_SETSERIALKEYS	シリアルキーアクセシビリティオプションのデータを設定します。
SPI_SETSHOWSOUNDS	サウンド表示モードを有効または無効にします。
SPI_SETSOUNDSENTRY	サウンドエントリアクセシビリティオプションのデータを設定します。
SPI_SETSTICKYKEYS	スティッキキーアクセシビリティオプションのデータを設定します。
SPI_SETTOGGLEKEYS	TOGGLEKEYS データを設定します。

Visual Basic で使用できるワークシート関数一覧

Visual Basic で使用できるワークシート関数の一覧です。これらの関数の詳細については、Microsoft Excel のオンラインヘルプを参照してください。

A

Acos、Acosh、And、Asin、Asinh、Atan2、Atanh、AveDev、Average

B

BetaDist、BetaInv、BinomDist

C

Ceiling、ChiDist、ChiInv、ChiTest、Choose、Clean、Combin、Confidence、Correl、Cosh、Count、CountA、CountBlank、、CountIf、Covar、CritBinom

D

DAverage、Days360、Db、DCount、DCountA、Ddb、Degrees、DevSq、DGet、DMax、DMin、Dollar、DProduct、DStDev、DStDevP、DSum、DVar、DVarP

E

Even、ExponDist

F

Fact、FDist、Find、FindB、FInv、Fisher、FisherInv、Fixed、Floor、Forecast、Frequency、FTest、Fv

G

GammaDist、GammaInv、GammaLn、GeoMean、Growth

H

HarMean、HLookup、HypGeomDist

I

Index、Intercept、Ipmt、Irr、IsErr、IsError、IsLogical、IsNA、IsNonText、IsNumber、Ispmt、IsText

K

Kurt

L

Large、LinEst、Ln、Log、Log10、LogEst、LogInv、LogNormDist、Lookup

M

Match、Max、MDeterm、Median、Min、MInverse、MIrr、MMult、Mode

N

NegBinomDist、NormDist、NormInv、NormSDist、NormSInv、NPer、Npv

O

Odd、Or

P

Pearson、Percentile、PercentRank、Permut、Phonetic、Pi、Pmt、Poisson、Power、Ppmt、Prob、Product、Proper、Pv

Q

Quartile

R

Radians、Rank、Rate、Replace、ReplaceB、Rept、Roman、Round、RoundDown、RoundUp、RSq

S

Search、SearchB、Sinh、Skew、Sln、Slope、Small、Standardize、StDev、StDevP、StEyx、Substitute、Subtotal、、Sum、SumIf、SumProduct、SumSq、SumX2MY2、SumX2PY2、SumXMY2、Syd

T

Tanh、TDist、Text、TInv、Transpose、Trend、Trim、TrimMean、TTest

U

USDollar

V

Var、VarP、Vdb、VLookup

W

Weekday、Weibull

Z

ZTest

A 練習問題の解答

Lesson1・・・・・・A1(Q&A-01.vbp)

フレキシブルグリッドコントロールの［AllowUserResizing］プロパティを「3－flexResizeBoth」に変更します。これで、ユーザーが自由にセルの幅と高さを変更できます。

プロパティ－MSFlexGrid1

MSFlexGrid1 MSFlexGrid

全体 | 項目別

(オブジェクト名)	MSFlexGrid1
(バージョン情報)	
(プロパティページ)	
AllowBigSelection	True
AllowUserResizing	3 - flexResizeBoth
Appearance	0 - flexResizeNone
BackColor	1 - flexResizeColumns
BackColorBkg	2 - flexResizeRows
BackColorFixed	3 - flexResizeBoth

ここをクリック

AllowUserResizing
ユーザーのマウス操作で行または列のサイズ変更を行うことができるかどうかを設定します。値の取得も可

Lesson2・・・・・・A1(Q&A-02.vbp)

小数点の合計計算を行わせるには、まず合計値を格納する変数「Store」を「Double」で宣言します。そして、合計計算を行う式で、データ型変換関数「CDbl」を使用します。この関数は、引数に与えられた文字列を、倍精度浮動小数点に変換します。

```
Dim Store As Double          ← データ型をDoubleに変える

On Error GoTo BACK
For i = StartCol To EndCol
    With MSFlexGrid1
        .Col = i
        Store = Store + CDbl(.Text)     ← CDbl関数を使う
```

簡単表計算ソフト

1.038　Input

	A	B	C	D	合計
1	1.23	0.012	1.038		2.28
2					
3					
4					
5					
6					
7					
8					

小数点の合計ができる

Lesson3・・・・・・A1(Q&A-03.vbp)

プログラムのポイントとしては、［ファイル名を付けて保存］ダイアログボックスに入力されたファイル名が存在するかどうかをDir関数でチェックし、存在すれば上書き確認のメッセージボックスを表示するようにします。

①まず、変数を追加します。変数「Ret」はDir関数の戻り値を受け取ります。「Msg」はメッセージボックスに表示するメッセージを格納する変数で、「SelectBtn」はメッセージボックスでクリックされたボタンの種類を格納する変数です。

```
Dim Ret As String
Dim Msg As String
Dim SelectBtn As Integer
```

②メッセージボックスに表示するメッセージを作成します。Chr関数と組み合わせて、2行のメッセージにします。

```
Msg = "すでに同じファイルが存在します。" & Chr(10) & "上書き保存しますか"
```

③ファイル名を変数GetFileNameに格納した後に、Dir関数を実行します。この関数は、引数にファイル名をフルパスで記述すると、ファイルが存在すればそのファイル名を返してくる関数です。ファイルが存在しなければ、空の文字「""」を返してきます。この関数の引数に、［ファイル名を付けて保存］ダイアログボックスで入力されたファイル名を与え、存在するかどうかを確認します。戻り値は変数「Ret」に格納します。

```
Ret = Dir(GetFileName)
```

④続いて、Ifステートメントでこの戻り値「Ret」の内容を判断します。もし、Dir関数の戻り値が空でなければ、メッセージボックスを表示します。メッセージボックスは、［はい］と［いいえ］の2つのボタンを持ったメッセージボックスを使用します。
Dir関数の戻り値が空であれば、存在しないファイル名なので、If処理を終了し、そのままデータの書き込み処理を行います。

```
If Ret <> "" then
    SelectBtn = MsgBox(Msg, vbYesNo, "ファイルの上書き確認")
```

⑤表示したメッセージボックスの、クリックされたボタンの種類を変数「SelectBtn」で受け取り、Ifステートメントで種類を判断します。そして、［いいえ］ボタンがクリックされれば、データの書き込み処理はせずに、Closeステートメントでファイルをクローズして、ラベル「BACK」にジャンプしプロシージャを終了します。

```
If SelectBtn = vbNo Then
    Close #1
    GoTo BACK
End If
```

⑥［はい］ボタンがクリックされれば、そのままデータ書き出しの処理を行います。データファイルは、新しいデータにすべて上書きされます。

Lesson4……A1(Q&A-04.vbp)

FreeFile関数は、現在未使用になっているファイル番号を返す関数です。この関数を使ってファイル番号を取得すれば、ファイル番号が重複してファイルが開けない、という事態を回避できます。使い方はいたって簡単で、まずFreeFile関数が返してくるファイル番号を受け取る変数を、整数型で宣言します。

```
Dim FileNo As Integer
```

次に、FreeFile関数を実行し、返してきたファイル番号をOpenステートメントに渡します

```
FileNo = FreeFile
Open GetFileName For Random As FileNo Len = DataByte
```

Lesson5……A1(Q&A-05.vbp)

①Visual Basicのツールパレットに、新しいコントロールを追加します。ツールパレットのショートカットメニュー［コンポーネント］を選び、［Microsoft Windows Common Controls 6.0］にチェックを付け、［OK］ボタンをクリックします。

ここをチェック

ここをクリック

②新しいフォームに、イメージリストコントロールを配置し、「32×32」のサイズでアイコンイメージを４つ組み込みます。［キー］は好きな文字列を設定してください。ここでは、「flag1」から「flag4」まで設定しました。

ここをクリック

キーを設定する

ここをクリックしてアイコンイメージを組み込む

③ツールバーコントロールをフォームに配置し、「イメージ」にイメージリストコントロールを設定します。
ボタンを４つ組み込んで、「キー」と「イメージ」を設定します。イメージリストコントロールのイメージサイズを「32 × 32」にしているので、Lesson5 で作成したツールバーよりも大きいサイズのボタンができあがります。

Lesson6……A1(Q&A-06.vbp)

①まず、フォームの Load イベントプロシージャに、[Caption] プロパティを設定するコードを記述します。

```
Private Sub Form_Load()
    Form1.Caption = "新規文書"
End Sub
```

また、[新規作成] メニューがクリックされたときにも、この処理が行われるようにコードを追加します。

```
Private Sub IDM_NEW_Click()
    RichTextBox1.TextRTF = ""
    Form1.Caption = "新規文書"

End Sub
```

②リッチテキストボックスコントロールの内容を保存したときは、「SaveFile」メソッドに渡した変数「Fname」を、そのままフォームの［Caption］プロパティに代入します。これで、保存したファイル名とそのパス名が、フォームのキャプションに表示されます。。

```
RichTextBox1.SaveFile Fname, rtfRTF
Form1.Caption = Fname
```

③ファイルを読み込んだ場合も、同じく「LoadFile」メソッドに渡した変数「Fname」を、そのままフォームの［Caption］プロパティに代入します。これで、保存したファイル名とそのパス名が、フォームのキャプションに表示されます。

```
RichTextBox1.LoadFile Fname, rtfRTF
Form1.Caption = Fname
```

これで、すべての場合で、現在編集している文書の名前がフォームに表示されるようになります。

Lesson7・・・・・・A1(Q&A-07.vbp)

印刷を実行するプロシージャ「Printout」を、コモンダイアログボックスコントロールを使うように書き換えます。

まず、コモンダイアログボックスコントロールの［Flags］プロパティに、定数「cdlPDReturnDC」を設定します。コモンダイアログボックスコントロールを使って印刷する場合は、「SelPrint」メソッドの引数に「Printer」オブジェクトではなくコモンダイアログボックスコントロールの［hDC］プロパティを、デバイスコンテキストとして設定する必要があります。［Flags］プロパティに定数「cdlPDReturnDC」を設定すると、［hDC］プロパティにデバイスコンテキストが格納されます。

また、リッチテキストボックスコントロールの［SelLength］プロパティを使うと、現在選択状態にある文字列の長さを知ることができます。もし、このプロパティの値が「0」であれば、まったく選択されていないことになります。これをIfステートメントで判断して、選択されていれば［Flags］プロパティに定数「cdlPDSelection」を、選択されていなければ定数「cdlPDAllPages」を設定して、印刷範囲を指定します。

あとは、コモンダイアログボックスコントロールの「ShowPrinter」メソッドを実行して［印刷］ダイアログボックスを表示し、［OK］ボタンがクリックされればリッチテキストボックスコントロールの「SelPrinter」を実行します。この時のデバイスコンテキストには、コモンダイアログボックスコントロールの［hDC］プロパティを設定することに注意してください。

また、［印刷］ダイアログボックスの［キャンセル］ボタンがクリックされるとエラーが発生しますから、これを「On Error」ステートメントで把握してエラー処理を組み込んでおきます。

※ネットワークプリンタでプリンタスプーラからプリントアウトする場合、このコードでは正常に動作しない場合があります。シリアルケーブルで直接接続しているプリンタでお試しください。

```
Private Sub Printout()
    CommonDialog1.Flags = cdlPDReturnDC
    If RichTextBox1.SelLength = 0 Then
        CommonDialog1.Flags = cdlPDAllPages
    Else
        CommonDialog1.Flags = cdlPDSelection
    End If

    CommonDialog1.ShowPrinter
    On Error GoTo FINISH
    RichTextBox1.SelPrint CommonDialog1.hDC
    Exit Sub

FINISH:
End Sub
```

Lesson8……A1(Q&A-08.vbp)

①新しいプロジェクトを用意して、ツールパレットにコンポーネント「Microsoft Windows Common Controls 6.0」を追加します。また、付録CD-ROMの「Material」フォルダにある「SMILEY.ICO」を使用します。
②フォームにイメージコントロールを配置し、［Picture］プロパティに「SMILEY.ICO」を指定します。
③スライダーコントロールをフォームに配置します。
④［Max］プロパティを「100」に、［Min］プロパティを「1」に設定します。
⑤スライダーコントロールの「Change」イベントプロシージャにコードを作成します。

```
Private Sub Slider1_Change()
    Image1.Left = Slider1.Value * 30
End Sub
```

このコードは、スライダーコントロールの移動量を［Value］プロパティから取り出し、30倍にしてイメージコントロールの［Left］プロパティに代入しています。これで、スライダーコントロールの移動量に伴って、イメージコントロールの左右方向の表示位置が変化します。

Lesson9……A1(Q&A-09.vbp)

ファイル名を検索する関数に、Dir関数があります。この関数は、引数にファイル名をフルパスで指定すると、そのディレクトリを検索して一致するファイルがあるかどうかを調べます。そして、最初に一致するファイルがあるとそのファイル名を返し、一致するファイルがなければ「""」（空白）を返してきます。また、一度Dir関数を実行したあとに、

引数を付けずにDir関数を実行すると、再び同じファイル名で検索を開始します。
検索するファイル名にはワイルドカード文字（*や?）が使用できるので、Do...Whileループと組み合わせれば、たとえば拡張子が「exe」のファイルだけを検索することができます。
このプログラムは、これらの機能を組み合わせて、Windowsがインストールされているフォルダのすべてのファイルを検索し、ループの中にファイル数を数える変数を組み込んで、個数を把握する処理を組み立てています。

●フォームの作成

①フォームにコマンドボタンとプログレスバーコントロールを配置します。

②プログレスバーコントロールの［Max］プロパティを「500」に設定します。
③コマンドボタンの［Caption］プロパティを「検索」に変えます。

●コードの作成

①コマンドボタンのClickイベントプロシージャに、検索の処理を作成します。検索したファイル名を格納する変数と、ファイル数を数える変数を宣言します。

```
Dim MyName As String
Dim i As Integer
```

②Dir関数で1回目の検索を実行します。引数にはWindowsがインストールされているディレクトリと、すべてのファイル（この場合は標準ファイルで隠しファイルとシステムファイルは除く）を検索するために、ファイル名にワイルドカードを使用しています。一致したときに返してくるファイル名は、変数「MyName」に格納します。また、万一「500」以上のファイルが見つかったときにエラーをスキップするように「On Error Resume Next」ステートメントを挿入しておきます。

```
MyName = Dir("c:¥windows¥*.*")
On Error Resume Next
```

また、この時点ですでに１つファイルが見つかっていますから、変数「i」に１を代入しておきます。

```
i = 1
```

③続いて、Do...While ループで２回目の Dir 関数を実行します。この関数では引数を指定しませんので、最初の引数「c:¥windows¥*.*」を使って次の該当ファイルの検索が行われます。もしファイルが見つかれば変数「MyName」にファイル名が返ってきます。

```
Do
    MyName = Dir
```

④ファイル名があれば、変数「i」の値を１つ増加し、その値をプログレスバーコントロールの［Value］プロパティに格納します。これで、検索状況をプログレスバーの変化でユーザーに知らせることができます。

```
i = i + 1
ProgressBar1.Value = i
```

⑤一致するファイルが見つからなければ、変数「MyName」には「""」が返されるので、その時点でループが終了するように、While の後の条件を設定しています。

```
Loop While MyName <> ""
```

⑥ファイルの総数がわかったら、プログレスバーコントロールの［Value］プロパティに Max プロパティの値を代入し、プログレスバーの表示を最大値にします。

```
With ProgressBar1
        .Value = .Max
```

そして、変数「i」の数字を見れば、全部でいくつのファイル名が見つかったのかがわかりますが、ループは最後の「""」が返されるまでまわっていますから、実際の総数は「i - 1」になります。
この数字をメッセージボックスで表示し、［OK］ボタンがクリックされれば、「0」に初期化し、次の動作に備えます。

```
MsgBox "ファイルの総数は " & i - 1 & "個です"
ProgressBar1.Value = 0
```

⑦以上で完成です。実際に動作させるとファイルの総数が表示されます。また、このディレクトリのフォルダを開いて、ファイル数が一致しているかどうかを確認します。
この「c:¥windows¥*.*」という検索条件では、フォルダに表示されるファイルの属性が「A」のファイルのみがカウントされ、隠しファイルとシステムファイル、フォルダと拡張子のないファイルは検索対象から外れます。拡張子が付いているファイルの総数をエクスプローラなどで実際に確認すると、プログラムの実行結果と一致しているはずです。

Lesson10 ････A1(Q&A-10.vbp)

このプログラムは、コマンドボタンがクリックされると、「■」の文字をステータスバーコントロールのパネルに追加していきます。

①まずフォームを用意し、コマンドボタンとステータスバーコントロールを配置します。

②ステータスバーコントロールの［プロパティページ］ダイアログボックスを表示し、［パネル］タブをクリックします。
［最小幅］の値を「3000」に設定します。

［パネル］をクリック

ここを「3000」に設定

③フォームのコードウィンドウを開き、宣言セクションに変数を２つ、モジュールレベルで宣言します。

```
Dim Mark As String
Dim i As Integer
```

④フォームのLoadイベントプロシージャで、変数「i」を初期化します。

```
Private Sub Form_Load()
    i = 0
End Sub
```

⑤コマンドボタンのClickイベントプロシージャに、ステータスバーコントロールの表示処理を作成します。この処理は、コマンドボタンが１回クリックされると、変数「Mark」に文字「■」を連結して格納します。つまり、コマンドボタンがクリックされるたびに、変数「Mark」の値は「■■■」と■マークが追加されて格納されます。これを、ステータスバーコントロールのパネルの［Text］プロパティに代入していきます。まず、変数「Mark」に文字「■」を代入します。

```
Private Sub Command1_Click()
    Mark = Mark & "■"
```

ワンポイントアドバイス

日本語入力システム（MS-IMEなど）をオンにして、キーボードから「しかく」と入力して変換すると、この■マークを呼び出せます。

⑥そして、この変数を「Panels(1)」の［Text］プロパティに代入します。

```
StatusBar1.Panels(1).Text = Mark
```

あわせて、何回コマンドボタンがクリックされたのかを把握するために、変数「i」の値も増加させておきます。

```
i = i + 1
```

⑦この変数「i」の値が「16」になったら、ステータスバーコントロールにはいっぱいに「■」マークが表示されていますので、これで終了させるために、Ifステートメントを使ってメッセージボックスを表示します。そして、変数「i」と「Mark」の値をリセットし、次の操作に備えます。

```
If i >= 16 Then
    MsgBox "処理は終了しました。"
    i = 0
    StatusBar1.Panels(1).Text = ""
    Mark = ""
End If
```

これで完成です。コマンドボタンをクリックするたびに、ステータスバーコントロールには■マークが追加されていきます。

Lesson11 ････A1(Q&A-11.vbp)

①ツールパレットに次のコンポーネントを追加します。
「Microsoft Windows Common Controls 6.0」
「Microsoft Windows Common Controls-2 6.0」
これで、アップダウンコントロールとツールバーコントロールが使用できます。

②アップダウンコントロールとツールバーコントロールをフォームに配置します。

③フォームのコードウィンドウを開き、宣言セクションに変数「i」を宣言します。

```
Dim i As Integer
```

④フォームのLoadイベントプロシージャで、この変数を「0」に初期化します。

```
Private Sub Form_Load()
    i = 0
End Sub
```

⑤アップダウンコントロールの「UpClick」イベントプロシージャに、ツールバーコントロールのボタンを追加する処理を組み込みます。ツールバーコントロールのボタンを追加するには、Addメソッドを使って、ツールバーコントロールのButtonsコレクションオブジェクトを追加します。追加したら、変数「i」の値を1つ増やしておきます。これは、何個のボタンを追加したのかがわかるようにするためです。

```
Private Sub UpDown1_UpClick()
    Toolbar1.Buttons.Add
    i = i + 1
End Sub
```

⑥アップダウンコントロールの「DownClick」イベントプロシージャに、ツールバーコントロールのボタンを削除する処理を作成します。
まず、何個のボタンが追加されているのかを、変数「i」の値で確認します。1つ以上のボタンが追加されていれば、「Remove」メソッドを使ってツールバーのButtonsコレクションを削除します。このとき、何番目のボタンを削除するのかを、「Remove」メソッドに指定します。これは、変数「i」を引き当てて行います。そして、ボタンを1つ削除したら、変数「i」の値を1つ減らしておきます。もし、1つのボタンも追加されていなければ、なんの処理も行いません。

```
Private Sub UpDown1_DownClick()
    If i >= 1 Then
        Toolbar1.Buttons.Remove (i)
        i = i - 1
    End If
End Sub
```

以上で完成です。

Lesson12 ····A1(Q&A-12.vbp)

プログラム起動時から、時刻表示を行わせるには、Timerイベントプロシージャで作成した処理とまったく同じ処理を、フォームのLoadイベントプロシージャで行えばいいのです。そこで、Timerイベントプロシージャに作成した処理をすべて、独自のプロシージャにしてしまい、Callステートメントでフォームの Load イベントプロシージャと、タイマーコントロールのTimerイベントプロシージャで呼び出せば、プログラム起動時から時刻表示が行われます。

Lesson13 ····A1(Q&A-13.vbp)

この処理は、Timerイベントプロシージャの中で行いますので、以下のコードを追加します。

①文字列を格納する変数を宣言します。

```
Dim JustTime As String
Dim JustMin As String
Dim JustSec As String
```

②ラベルコントロールに出力した、Time関数で取得した時刻を、一度変数「JustTime」に格納します。

```
JustTime = Label1.Caption
```

③そして、Mid関数とRight関数で分と秒の文字を取り出します。取り出した文字列は、それぞれ変数「JustMin」と「JustSec」に格納します。

```
JustMin = Mid(JustTime, 4, 2)
JustSec = Right(JustTime, 2)
```

④時刻がちょうどになったときというのは、分も秒も「00」になったときです。そこで、Ifステートメントを使用して、それぞれの文字が「00」であれば、Beepステートメントを実行してビープ音を鳴らすようにします。このとき、IfステートメントではAND演算子を使用して、2つの条件が重なったときだけBeepステートメントを実行するようにします。

```
If (JustMin = "00") And (JustSec = "00") Then
    Beep
End If
```

Lesson14 ････A1(Q&A-14.vbp)

①まず、ツールパレットに「Microsoft Common Dialog Control 6.0」を追加します。
②フォームにコモンダイアログコントロールを配置します。［Color］プロパティの値を「&H00FFFF00&」（シアン）に設定します。

コモンダイアログコントロールを配置する

シアンを設定

③［プロパティページ］ダイアログボックスを表示し、［フォント］タブの［フォント名］に「MS Pゴシック」を設定します。

「MS Pゴシック」を設定

④メニューエディタで、次のメニューを組み込みます。

キャプション	名前	表示
設定	IDM_SETUP	チェックを外す
フォント	IDM_FONT	チェックあり
文字色の設定	IDM_COLOR	チェックあり

⑤フォームのMouseDownイベントプロシージャに、右ボタンが押されたときに、[設定] メニューを表示するように、PopupMenuメソッドを実行します。

```
Private Sub Form_MouseDown(Button As Integer, _
    Shift As Integer, X As Single, Y As Single)
    If Button = vbRightButton Then
        PopupMenu IDM_SETUP
    End If
End Sub
```

⑥ [フォント] メニューのClickイベントプロシージャにコモンダイアログの [フォント] ダイアログボックスの処理を作成します。[Flags] プロパティに定数「cdlCFBoth」を設定し、「ShowFont」メソッドを実行します。ユーザーによって選択されたフォント名とサイズを、ラベルコントロールの各プロパティに代入します。

```
Private Sub IDM_FONT_Click()
    With CommonDialog1
        .Flags = cdlCFBoth
        .ShowFont
        Label1.Font.Name = .FontName
        Label1.Font.Size = .FontSize
    End With
End Sub
```

⑦ [文字色の設定] メニューのClickイベントプロシージャに、[色の設定] ダイアログボックスを表示する処理を作成します。選択された色は、そのままラベルコントロールの [ForeColor] プロパティに代入します。

```
Private Sub IDM_COLOR_Click()
    With CommonDialog1
        .ShowColor
        Label1.ForeColor = .Color
```

```
    End With
End Sub
```

⑧以上で完成です。

Lesson15 ････A1(Q&A-15.vbp)

Meキーワードを使うと、現在アクティブになっているオブジェクトを参照できます。たとえば、コマンドボタンのClickイベントプロシージャに、「Me」と入力して「.」を入力すると、コマンドボタンが組み込まれているフォーム（すなわち子フォーム）の、プロパティとメソッドのリストが表示されます。

子フォームのプロパティとメソッドのリスト

このように、プログラム実行時に常にアクティブな状態が入れ替わるオブジェクトのプロパティ等を操作したい場合は、「Me」キーワードを活用します。

ここでは、コマンドボタンがクリックされたときに、コマンドボタンが組み込まれている子フォームの［Caption］プロパティにアクセスしたいので、次のように記述します。

```
Me.Caption
```

あとは、この［Caption］プロパティの値を、表示するメッセージの文字列に設定すればできあがります。

また、子フォームの［Caption］プロパティに、そのフォームの名前を付けたい場合は、インスタンスを作成するプロシージャの中で、文字列の設定を行うコードを記述しておきます。

```
With AddForm
    .Show
    .Caption = "Add Form" & i
```

```
    .Label1.Caption = "このフォームは " & "Add Form" & i
End With
```

Lesson16 ····A1(Q&A-16.vbp)

ツールバーコントロール上のマウスポインタを変更するには、プロパティページとアイコンファイルを使用します。

①まず、使用するアイコンファイルを用意します。ここでは、付録CD-ROMの「material」フォルダにある「Finger.ico」ファイルを使用します。

このファイルを使用

②ツールバーコントロールの［プロパティページ］ダイアログボックスを開きます。そして、「マウスポインタ」の値をリストから選び、「99 – ccCustom」に設定します。

ここをクリック

③［ピクチャ］タブをクリックして、［参照］ボタンをクリックします。フォルダを選び、「Finger.ico」ファイルを指定します。

このファイルを指定

④［プレビュー］にアイコンファイルが表示されたら、［OK］ボタンをクリックします。

ここをクリック

これで、ツールバーコントロールにマウスポインタが置かれると、設定したアイコンイメージに変化します。

マウスポインタが指の形に変わる

Lesson17 ････A1(Q&A-17.vbp)

①インスタンスのフォームにアイコンを組み込むには、［Icon］プロパティに、イメージリストコントロールのアイコンを組み込みます。
イメージリストコントロールのピクチャを取得するには、「ListImages()」コレクションオブジェクトの［Picture］プロパティを使います。この値を、フォームの［Icon］プロパティに代入すれば表示されます。テキスト用のピクチャは、インデックスが「2」なので、この値を代入します。

```
.Icon = ImageList1.ListImages(2).Picture
```

このコードを、プロシージャ「OpenText」に組み込みます。

また、イメージ用のピクチャは、イメージリストコントロールのインデックス「3」のピクチャですから、同様に［Picture］プロパティの値を代入します。

```
.Icon = ImageList1.ListImages(3).Picture
```

② MDIフォームにステータスバーコントロールを組み込んで、プロパティページを表示します。パネルを1つ追加し、［スタイル］を［5 − sbrTime］に、［サイズの自動設定］を［1 − sbrSpring］に変更します。

Lesson18 ・・・・A1(Q&A-18.vbp)

①フォームにテキストボックスコントロールを配置します。あわせて、コマンドボタンのレイアウトなども修正します。場合によっては、ツールバーコントロールに入れ替えてもかまいません。

②テキストボックスコントロールの［Text］プロパティに、デフォルトのURLを設定しておきます。

③［Get］ボタンのClickイベントプロシージャを修正し、テキストボックスコントロールの［Text］プロパティからURLを取得するようにします。これで、テキストボックスに入力されたURLのHTMLファイルを、Executeメソッドに渡すことができます。

```
GetUrl = Text1.Text
Inet1.Execute GetUrl, "GET"
```

④インターネットトランスファーコントロールの「StateChanged」イベントプロシージャも修正します。テキストボックスコントロールに入力されたURLが正しくなかった場合のエラー処理を加えます。
指定したURLに対し、Executeメソッドの「GET」コマンドが実行できなかった場合は、「StateChanged」イベントプロシージャの引数「State」に、定数「icError」が設定されます。この値をIfステートメントで評価し、エラーであればメッセージボックスで表示します。ダウンロードに失敗していますから、リッチテキストボックスコントロールへの内容の表示も行いません。

```
ElseIf State = icError Then
    MsgBox "アクセスに失敗しました。URLを確認してください。"
```

表示されたメッセージ

Lesson19 ････A1(Q&A-19.vbp)

①フォームにプログレスバーコントロールを配置します。そして、［Max］プロパティを「10」に設定します。

プログレスバーコントロールを配置

「10」を設定

②フォームの宣言セクションに、モジュールレベルの変数「i」を宣言します。

```
Dim i As Integer
```

そして、フォームのLoadイベントプロシージャで初期化します。この変数を使って、プログレスバーコントロールの動作を設定します。

```
Private Sub Form_Load()
    i = 1
End Sub
```

③プログレスバーコントロールの操作は、インターネットトランスファーコントロールの「StateChanged」イベントプロシージャで行います。

インターネットトランスファーコントロールが、サーバに接続して応答を受信すると、引数「State」には「icReceivingResponse」が格納されます。この定数は、受信をしている間はつねに設定されます。したがって、このCase節でプログレスバーコントロールの［Value］プロパティの値を増やしていけば、受信中は絶えずプログレスバーが動いていることになります。そのために、プログレスバーコントロールの［Value］プロパティの値に、変数「i」と式を組み込んでいます。そして、この変数「i」の値が［Max］プロパティの値に達したら、変数を「0」に戻してあげれば、受信中は絶えず繰り返しプログレスバーが動いていることになります。

```
Case icReceivingResponse
    i = i + 1
    If i = 10 Then
        i = 0
    End If
    ProgressBar1.Value = i
```

④そして、ダウンロードが完了すれば、「State」には「icResponseCompleted」が格納されますから、その時点でプログレスバーの［Value］プロパティを「0」に戻し、表示を取り消します。

```
Case icResponseCompleted
    GetData = Inet1.GetChunk(2048, icString)
    RichTextBox1.Text = GetData
    MsgBox "ダウンロード完了"
    ProgressBar1.Value = 0
```

なお、このプロシージャは、「State」での分岐の数が増えたので、Ifステートメントから Select Caseステートメントに処理を変更しています。

Lesson20 ••••A1(Q&A-20.vbp)

ユーザーがリストビューコントロールに表示している項目の名称を変更すると、リストビューコントロールには「AfterLabelEdit」イベントが発生します。そこで、このイベントが発生したら（つまり、項目名が修正されたら）、［OK］と［キャンセル］の２つのボタンを持ったメッセージボックスを表示します。

次に、MsgBox関数の戻り値を把握し、［キャンセル］ボタンがクリックされたら、項目名の変更を取り消します。「AfterLabelEdit」イベントプロシージャの引数「Cancel」に「True」を設定すると、この項目名の変更を取り消し、変更前の項目名に戻してくれます。

```
Private Sub ListView1_AfterLabelEdit(Cancel As Integer, _
    NewString As String)
    Dim Ret As Integer
```

```
    Ret = MsgBox("名称を変更しますか?", vbOKCancel)
    If Ret = vbCancel Then
        Cancel = True
    End If
End Sub
```

Lesson21 ····A1(Q&A-21.vbp)

①ツールパレットにコモンダイアログコントロールを追加します。

②コモンダイアログコントロールをフォームに配置します。［プロパティページ］ダイアログボックスを開き、次のプロパティを設定します。

プロパティ	値
タイトル	ファイルの保存
初期ディレクトリ	c:¥

③リストビューコントロールの「ItemClick」イベントプロシージャを修正します。
まず、コモンダイアログの［filename］プロパティに、リストビューコントロールでクリックされたファイル名を代入します。これで、コモンダイアログボックスにこのフ

ァイル名が表示されます。また、変数「GetFile」にも、ファイル名を格納します。

```
Private Sub ListView1_ItemClick(ByVal Item As _
    ComctlLib.ListItem)
    CommonDialog1.Filename = Item.Text
    GetFile = Item.Text
End Sub
```

クリックしたファイル名が
表示されている

④［Download］ボタンのClickイベントプロシージャを修正します。変数「Flag」の値はそのままで、コモンダイアログボックスを「ShowSave」メソッドで表示します。そして、コモンダイアログボックスの［filename］プロパティから、フルパスで転送先のファイル名を取得し、「GET」コマンドの記述を作成し、Executeメソッドを実行します。

```
Private Sub DownBtn_Click()
    Flag = False
    With CommonDialog1
        .ShowSave
        GetFile = "GET " & GetFile & " " & .Filename
    End With

    ListView1.SetFocus
    Inet1.Execute GetUrl, GetFile
End Sub
```

Lesson22 ････A1(Q&A-22.vbp)

①［新しいブラウザの起動］ボタンがクリックされた時点で、このボタン自身を「無効」にします。

```
RunBtn.Enabled = False
```

②［ブラウザを閉じる］ボタンをクリックすると、［新しいブラウザの起動］ボタンを有

効にし、［ブラウザを閉じる］ボタンと［ブラウザの表示切り替え］ボタンを無効にします。

```
ViewBtn.Enabled = False
CloseBtn.Enabled = False
RunBtn.Enabled = True
```

これで、インスタンスは1つしか作成できません。

なお、ここでは［ブラウザを閉じる］ボタンをクリックすると、［Visible］プロパティを操作するのではなく、「Quit」メソッドを使用しています。このメソッドを使うと、インスタンスを破棄してブラウザを終了します。

［新しいブラウザの起動］ボタンが無効になってインスタンスを作成できない

Lesson23 ････A1(Q&A-23.vbp)

①フォームにフレームコントロール、コマンドボタン、ラベルコントロールを配置します。

追加したコントロール

②インターネットエクスプローラが現在表示しているページのURLを取得するには、「LocationURL」プロパティの値を参照します。この値を、ラベルコントロールに出力すれば、現在表示しているページのURLを知ることができます。

```
Label1.Caption = Ie.LocationURL
```

③このコードを、追加したコマンドボタンのClickイベントプロシージャに組み込みます。

```
Private Sub Command1_Click()
    Label1.Caption = Ie.LocationURL
End Sub
```

これで、このボタンをクリックすれば、現在表示しているページのURLが、ラベルコントロールに表示されます。

④そして、［ブラウザを閉じる］ボタンの処理に、ラベルコントロールの文字をクリアにする処理を作成します。インターネットエクスプローラが閉じられれば、URLの表示も取り消します。

```
Label1.Caption = ""
```

ボタンをクリックすると、表示しているページのURLが表示される

Lesson24 ････A1(Q&A-24.vbp)

サーバからの転送が完了したときに、WebBrowserコントロールに発生するイベント「DownloadComplete」のプロシージャに、［LocationURL］プロパティの内容をフォームの［Caption］プロパティに設定する処理を作成します。

これで、現在表示しているページのURLが、フォームのキャプションに表示されます。

```
Private Sub WebBrowser1_DownloadComplete()
    Label1.Caption = "表示完了"
    Form1.Caption = WebBrowser1.LocationURL
End Sub
```

現在表示しているページのURLが表示される

この処理は、ローカルコンピュータのファイルを表示する場合は、うまく動作しない場合があります。

Lesson24 ････A2(Q&A-24.vbp)

コンボボックスコントロールの「KeyDown」イベントを使用すると、テキスト入力エリアでEnterキーが押されたときの処理が作成できます。

このイベントプロシージャは、引数「KeyCode」に、押されたキーのコードを格納します。Enterキーは、コード「13」ですから、Ifステートメントでこれを評価し、「13」であれば、WebBrowserコントロールの「Navigate」メソッドを実行します。

```
Private Sub Combo1_KeyDown(KeyCode As Integer, Shift As Integer)
    If KeyCode = 13 Then
        On Error Resume Next
        WebBrowser1.Navigate Combo1.Text
    End If
End Sub
```

Lesson24 ････A3(Q&A-24.vbp)

①フォームにコマンドボタンを配置し、[Caption] プロパティに「SaveAs」の文字列を設定し、[Enabled] プロパティを「False」にします。また、オブジェクト名を「SaveAsBtn」に変更します。

配置してプロパティを設定したコマンドボタンコントロール

② [Go] ボタンのClickイベントプロシージャで、[SaveAs] ボタンの [Enabled] プロパティを「True」に設定し、ボタンを有効にします。

```
Private Sub GoBtn_Click()
    SaveAsBtn.Enabled = True
    On Error Resume Next
    WebBrowser1.Navigate Combo1.Text
End Sub
```

③ [SaveAs] ボタンのClickイベントプロシージャに、WebBrowserコントロールの「ExecWB」メソッドを実行するコードを作成します。
このメソッドは、いくつかの処理を実行するメソッドで、最初の引数に実行するコマンドの種類を指定します。このコマンドは、コードウィンドウに入力する際に、引数リストが表示されますから、ここから選びます。

QA24 - Form1 (コード)
SaveAsBtn
Click
```
Private Sub SaveAsBtn_Click()
    On Error Resume Next
    WebBrowser1.ExecWB |
End Sub
```
Exec
OLECMDID_REDO
OLECMDID_REFRESH
OLECMDID_SAVE
OLECMDID_SAVEAS
OLECMDID_SAVECOPYAS
OLECMDID_SELECTALL
OLECMDID_SETDOWNLOADSTATE
DEXECOPT, [pvaIn],

引数のリストから
コマンドを選ぶ

WebBrowserコントロールに表示しているドキュメントを保存するには、第一引数に「OLECMDID_SAVEAS」を、第二引数に「OLECMDEXECOPT_DODEFAULT」を指定します。

```
Private Sub SaveAsBtn_Click()
    On Error Resume Next
    WebBrowser1.ExecWB OLECMDID_SAVEAS, OLECMDEXECOPT_DODEFAULT
End Sub
```

これで、[SaveAs] ボタンをクリックするだけで、[名前を付けてファイルを保存] ダイアログボックスが表示され、ファイルに保存できるようになります。ダイアログボックスの [キャンセル] ボタンがクリックされると、「ExecWB」メソッドがエラーになりますので、「On Error Resume Next」ステートメントを加えておきます。

ここをクリックすると…

［名前を付けてファイルを保存］ダイアログボックスが表示される

Lesson25 ····A1(Q&A-25.vbp)

①フォームにコマンドボタンを追加し、［Caption］プロパティを「Windowsのディレクトリ」に変更します。

コマンドボタンを追加して［Caption］プロパティを変更

②標準モジュール（Module1）に、API関数「GetWindowsDirectory」を、Declareステートメントを使って宣言します。
この API は、先に使用した API「GetComputerName」と同じような形態の引数を持ち、実行すると引数「lpBuffer」に、Windows がインストールされているディレクトリ名を格納します。

```
Declare Function GetWindowsDirectory Lib "kernel32" _
                        Alias "GetWindowsDirectoryA" _
                        (ByVal lpBuffer As String, _
                        ByVal nSize As Long) As Long
```

③フォームの［Windowsのディレクトリ］ボタンのClickイベントプロシージャで、このAPIを実行する処理を記述します。変数「Pathname」は、20バイトの固定長で宣言します。そして、「GetComputerName」を実行し、変数「Pathname」の値をラベルコントロールに出力します。

```
Private Sub Command2_Click()
    Dim Pathname As String * 20

    GetWindowsDirectory Pathname, 20
    Label1.Caption = Pathname
End Sub
```

Form1
使用しているコンピュータ名
Windowsのディレクトリ
I:¥WIN98

ここをクリックすると...

Windowsが保存されているディレクトリが表示される

Lesson26 ····A1(Q&A-26.vbp)

ウィンドウのキャプションを点滅させる機能は、Visual Basicには組み込まれていません。そこで、API関数をプログラムに組み込んで使用します。

① APIビューワを起動します。［ファイル］メニューの［データベースファイルの読み込み］を選択して、「WIN32API.MDB」を開きます。

Jet データベースを開く
ファイルの場所(I): Winapi
Win32api.MDB
ファイル名(N): Win32api.MDB
ファイルの種類(T): MDB ファイル(*.MDB)
開く(O)
キャンセル

ここをクリック

② API関数「FlashWindow」を［有効な項目］から選び、Declare宣言文を、Visual Basicの標準モジュールにコピーします。

③フォームに、タイマーコントロールを１つ配置し、「Interval」プロパティを「0」にしておきます。また、コマンドボタンを１つ配置し、［Caption］プロパティを「リセット」に変更します。

Form1

現在のコンピュータ名

使用しているコンピュータ名

新しいコンピュータ名の設定

配置してプロパティを変更したタイマーコントロール

④ API関数「FlashWindow」は、2つの引数を持ちます。1つは、「ウィンドウハンドル（hwnd）」と呼ばれる、ウィンドウ1つ1つを識別する値を持ちます。もう1つの引数には、点滅を有効にするのか無効にするのかを決める、論理値を設定します。

```
Declare Function FlashWindow Lib "user32" _
    (ByVal hwnd As Long, _
    ByVal bInvert As Long) As Long
```

⑤このプログラムではフォームのキャプションを点滅させますから、ウィンドウハンドルはフォームになります。フォームのウィンドウハンドルを取得するには、［hwnd］プロパティを使用します。このプロパティは、フォームのハンドルを返すプロパティです。これを、「FlashWindow」の引数に設定し、第二引数に「True」を設定すると、フォームのキャプションが1回点滅します。このコードを、タイマーコントロールの「Timer」イベントに記述すれば、タイマーが動作するたびにフォームのキャプションが点滅します。

```
Private Sub Timer1_Timer()
    FlashWindow Form1.hwnd, True
End Sub
```

ただし、タイマーコントロールの［Interval］プロパティを「0」にしてあるので、この時点ではまだ点滅は行われません。

⑥このプログラムでは、フォームがクリックされると、キャプションの点滅が開始するようにしますので、フォームのClickイベントプロシージャに、タイマーコントロールの［Interval］プロパティを「500」（0.5秒）に設定します。

```
Private Sub Form_Click()
  Timer1.Interval = 500
End Sub
```

⑦フォームがダブルクリックされた時点で、点滅を止めますので、フォームのDblClickイベントプロシージャで、タイマーコントロールの［Interval］プロパティを「0」にします。

```
Private Sub Form_DblClick()
    Timer1.Interval = 0
End Sub
```

⑧最後に、配置したコマンドボタン「リセット」のClickイベントプロシージャで、「FlashWindow」の第二引数に「False」を設定して実行します。これで、「FlashWindow」関数によるキャプションの点滅をリセットします。

```
Private Sub Command1_Click()
    FlashWindow Form1.hwnd, False
End Sub
```

Lesson27 ････A1(Q&A-27.vbp)

①まず、フォームにラベルコントロールを配置し、［Caption］プロパティを「使用済みメモリ」に変えます。そして、値を表示するラベルコントロールを、コピー＆貼り付けで追加します。このラベルコントロールは、「Label5(4)」になります。

②フォームのコードウィンドウを表示し、コマンドボタンのClickイベントプロシージャを修正します。まず、変数「TtlMem」をLong型で宣言します。

```
Dim TtlMem As Long
```

次に、この変数に「dwTotalPhys」から「dwAvailPhys」を差し引いた値を代入します。総メモリから使用可能なメモリを引けば、使用中のメモリ量が算出できます。

```
TtlMem = .dwTotalPhys - .dwAvailPhys
```

③この値を追加したラベルコントロールに出力します。このとき変数「TtlMem」の値を「1000000」（100万）で割れば、メガバイト単位に換算できますが、これだと浮動

小数点になってしまいます。そこで、「CInt」関数を使って、小数点を丸めて整数に変換します。こうすれば、たとえば使用可能なメモリ量が「47210496」バイトだとすれば、100万で割った「47,210496」を丸めて「47MB」と表示できます。

小数点で表示するとこうなる

CInt関数で小数点を丸めて表示

Lesson28 ････A1(Q&A-28.vbp)

すでに使用したAPI関数とテクニックを組み合わせていきます。

①フォームにラベルコントロールを配置し、大きく広げてプロパティを設定します。

プロパティ	プロパティ値
Alignment	2－中央揃え
BorderStyle	1－実線
Caption	（なし）

配置してプロパティを設定したラベルコントロール

②プロジェクトに標準モジュールを追加し、4つのAPI関数をDeclareステートメントで宣言します。

```
GetComputerName
GlobalMemoryStatus
GetSystemInfo
GetVersionEx
```

また、これらのAPI関数で使用する構造体も、ユーザー定義のデータ型として宣言します。

```
OSVERSIONINFO
SYSTEM_INFO
MEMORYSTATUS
```

③プログラムは、フォームが表示された時点で、各種の情報を取得し表示するようにしますので、フォームのLoadイベントプロシージャに処理を作成します。処理の内容は、これまでそれぞれのレッスンで紹介したものと同じなので、そちらをもう一度参照してください。

④メモリの使用状況の処理では、

```
CInt(.dwTotalPhys / 1000000) & "MB"
```

という式を使って、表示をメガバイト単位にしています。

⑤また、このプログラムでは1つのラベルコントロールに、すべての情報をまとめて表示するので、文字列を作成するために「Msg(8)」という配列変数を使っています。メッセージを配列の変数に格納しておくと、For...Nextループで一気に文字列連結ができるようになり、処理が簡単になります。改行の挿入に、「Chr(10)」関数を使用しているのも、ポイントです。
次のコードは、[Caption]プロパティの値に、For...Nextループで次々と文字列を連結して格納します。配列のインデックスとループのカウンタ変数の値をきちんと合わせておく必要があります。

```
For i = 0 To 8
    Label1.Caption = Label1.Caption & Chr(10) & Chr(10) _
                                    & Msg(i)
Next
```

Lesson29 ････A1(Q&A-29.vbp)

①フォームに配置したラベルコントロールを、コンボボックスコントロールに入れ替えます。

ラベルコントロールを削除し、コンボボックスコントロールを配置する

②コンボボックスコントロールの［List］プロパティに、いくつかのビットマップファイルの名前を登録します。

［List］プロパティにビットマップファイル名を登録する

それに伴い、［壁紙の選択］ボタンでラベルコントロールに出力していたファイル名の処理を、コンボボックスコントロールの［Text］プロパティへの代入式に入れ替えます。

```
Combo1.Text = Fname
```

③コンボボックスコントロールのリストからファイル名が選択されると「Click」イベントが発生します。そこで、このイベントプロシージャで、選択されたファイル名を変数「Fname」に格納する処理を行います。この処理は、独自のプロシージャ「GetList」を作成して、これを呼び出します。

```
Private Sub Combo1_Click()
    GetList
End Sub
```

④また、テキストエリアに文字が入力された場合も同様に、入力されたファイル名を変数「Fname」に格納するようにします。この場合は、Enterキーが押された時点で入力が確定されたことにしますので、「KeyDown」イベントプロシージャでは、「KeyCode」

を使って押されたキーの種類を判別し、Enterキーが押されたときだけ、プロシージャ「GetList」を呼び出し、ファイル名を変数に格納するようにします。

```
Private Sub Combo1_KeyDown(KeyCode As Integer, Shift As Integer)
    If KeyCode = 13 Then
        GetList
    End If
End Sub
```

⑤コンボボックスコントロールの操作でファイル名を取得する独自のプロシージャ「GetList」は、コンボボックスコントロールの［Text］プロパティの値を変数「Fname」に格納する処理を行います。

```
Sub GetList()
    On Error GoTo FAIL
    Fname = Combo1.Text
    Image1.Picture = LoadPicture(Fname)
    SetBtn.Enabled = True
    Exit Sub
```

もし、ファイル名が空欄だったり、存在しないファイル名が入力された場合はエラーが発生しますから、「On Error GoTo」ステートメントで処理をラベル「FAIL」までジャンプさせます。

⑥ラベル「FAIL」では、イメージコントロールの［Picture］プロパティを空欄にして、表示を取り消します。また、［設定］ボタンを無効にして、エラーのメッセージを表示します。
メッセージボックスでは、Errオブジェクトの［Number］プロパティでエラー番号を取得し、Error関数でエラー番号に対応したエラーメッセージを表示します。そして、再入力に備えて、コンボボックスコントロールにフォーカスを移します。

```
FAIL:
    Image1.Picture = LoadPicture("")
    SetBtn.Enabled = False
    MsgBox Error(Err.Number)
    Combo1.SetFocus
End Sub
```

壁紙チェンジャー
壁紙の設定
壁紙の選択
c:¥OPM.jpg
設定
Q&A-29
ファイルが見つかりません: 'c:¥OPM.jpg'
OK
壁紙の削除
終了

エラー番号に対応したエラーメッセージを表示

Lesson30 ････A1(Q&A-30.vbp)

Time関数を使用すると、現在の時刻を取得できますので、これを使用して設定した時間になれば、API関数「SystemParametersInfo」を実行するようにします。ただし、プログラムが常に現在の時刻を把握する必要がありますので、タイマーコントロールの「Timer」イベントの中で、Time関数による現在時刻の把握を行っておく必要があります。

①タイマーコントロールの［Interval］プロパティを「1000」（1秒）に設定します。

②タイマーコントロールのTimerイベントプロシージャを修正します。
まず、現在時刻を格納する変数「NowTime」を宣言します。そして、この変数に、Time関数が戻してくる現在の時刻を格納します。

③しかし、Time関数は、システムに設定されている時刻表示のフォーマットで、現在時刻を返してきます。たとえば、筆者のコンピュータでは、「午後 3:24:43」という形式です。これだと、時刻の数字を把握しづらいので、いったん「Format」関数を使用して、「15:24:43」という時刻書式に変換してから、変数「NowTime」に格納します。

```
NowTime = Format(Time, "hh:mm:ss")
```

④この取得した時刻を、Ifステートメントで判断します。設定した時刻が「15:00:00」であれば、壁紙を変更する処理を一気に実行します。このとき、必ず秒単位まできちんと確認してください。

⑤Timerイベントは、［Interval］プロパティを「1000」に設定していますので、1秒ごとに時刻のチェックが行われ、設定した時間になると壁紙を変更します。

Lesson31 ････A1(Q&A-31.vbp)

API関数「GetDriveType」を使用すると、ドライブのタイプを返してきます。これを、フレームコントロールのキャプションで表示します。

API関数「GetDriveType」は、引数「nDrive」にドライブ名を指定して実行すると、そのドライブのタイプを定数で返してきます。

```
Declare Function GetDriveType Lib "kernel32" _
        Alias "GetDriveTypeA" _
        (ByVal nDrive As String) As Long
```

返ってくる定数は、Visual Basicの形式では、次のように再定義されます。

```
Public Const DRIVE_REMOVABLE = 2
Public Const DRIVE_FIXED = 3
Public Const DRIVE_REMOTE = 4
Public Const DRIVE_CDROM = 5
Public Const DRIVE_RAMDISK = 6
```

この戻り値を使って、Select Caseステートメントで分岐して文字列に変換し、フレームコントロールの［Caption］プロパティに設定します。

Select Caseステートメントでは、直接関数の実行文を条件として記述すると、自動的に戻り値を分岐の条件にします。また、Case節では、戻り値の定数をそのまま設定すれば、わかりやすい記述になります。

```
Dim Drvtype As String
```

```
Select Case GetDriveType(lpRootPathName)
    Case DRIVE_REMOVABLE
        Drvtype = "Removable Disk"

    Case DRIVE_FIXED
        Drvtype = "Fixed Hard Disk"

    Case DRIVE_REMOTE
        Drvtype = "Remote Network Drive"

    Case DRIVE_CDROM
        Drvtype = "CD-ROM Drive"

    Case DRIVE_RAMDISK
        Drvtype = "RAM Disk"
End Select
```

得られたドライブのタイプを、フレームコントロールの［Caption］プロパティに設定する際に、ドライブリストボックスの［Drive］プロパティの値と結合すれば、ドライブ名やボリュームラベルを表示に使用できます。

```
Frame1.Caption = "Drive " & Drive1.Drive & " " & Drvtype
```

Lesson32 ････A1(Q&A-32.vbp)

Excelの起動時にサイズを指定するには、Excelでいう［Application］オブジェクトの［Width］と［Height］プロパティに、サイズをポイント単位で指定します。また、デスクトップの表示位置を指定するには、同じく「Application」オブジェクトの［Top］と［Left］プロパティに、位置を設定します。

この場合は、Excelのインスタンスのオブジェクトを格納しているオブジェクト変数を指定して、プロパティに値を設定します。

```
.Height = 500
.Width = 700
.Left = 10
.Top = 10
```

Lesson33 ････A1(Q&A-33.vbp)

付録CD-ROMの「Q&A」フォルダにあるExcelのブック「photoalbum.XLS」と「F1.XLS」を、使用しているコンピュータのプログラムを作成するフォルダにコピーしてください。

①フォームにコマンドボタンを1つ配置し、［Caption］プロパティを「ウィンドウ名を

表示」変更します。

配置してプロパティを変更したコマンドボタンコントロール

②このボタンのClickイベントプロシージャに、ウィンドウ名を表示するコードを作成します。まず、表示するウィンドウ名を格納する変数を宣言します。

```
Dim msg As String
```

次に、ExcelのApplicationオブジェクトの［ActiveWindow］プロパティを使用して、Excelで最前面（アクティブ）になっているウィンドウを把握します。そして、このWilndowオブジェクトの［Caption］プロパティの値を参照すると、現在アクティブになっているウィンドウ名を取得できます。

```
msg = XLSBook.Application.ActiveWindow.Caption
```

③あとは、文字列を加えてメッセージに加工して、メッセージボックスで表示します。

```
MsgBox "現在のウィンドウ名 : " & msg
```

④ただし、このままでは、［新しいブックを起動］ボタンと［ファイルからExcelを起動］ボタンのClickイベントプロシージャでは、Excelを起動したあとに、

```
Set XLSBook = Nothing
```

とオブジェクト変数を解放してしまっています。そのため、他の処理で作成したExcelオブジェクトを参照できなくなってしまいます。そこで、これらのプロシージャからオブジェクト変数の解放コードを削除し、フォームの「Unload」イベントプロシージャで一括して解放するようにしています。

```
Private Sub Form_Unload(Cancel As Integer)
    Set XLSBook = Nothing
End Sub
```

⑤これで完成です。Excelを起動し、［ウィンドウ名を表示］ボタンをクリックします。次にウィンドウを切り替えて、再度このボタンをクリックすると、ウィンドウ名も切り替わります。

現在のウィンドウ名を表示

ウィンドウを切り替えると、ウィンドウ名も変わる

Lesson34 ••••A1(Q&A-34.vbp)

①フォームにコマンドボタンを配置し、［Caption］プロパティを「Min」に、オブジェクト名を「MinBtn」に変更します。

配置してプロパティを変更したコマンドボタンコントロール

②［Ave］ボタンの横のラベルコントロールをコピーして、［Min］ボタンの横に貼り付けます。

これをコピーして...

ここに貼り付ける

③このボタンのClickイベントプロシージャに、ワークシート関数のMin関数を実行するコードを作成します。Min関数は、引数にセル範囲を指定すると、その範囲内のデータのうち、もっとも小さい値を返す関数です。ここでは、セル範囲にRangeオブジェクトを指定して実行し、戻り値をラベルコントロールに出力します。

```
Label1(2).Caption = AppXLS.WorksheetFunction.Min(CheckRange)
```

入力したデータの最小値が表示される

④Enterキーを押すだけでリストへ連続入力するために、ここでは引数「KeyAscii」を使っておきましょう。Enterキー（KeyAscii = 13）が押されると「InputBtn_Click」プロシージャに処理を渡します。最後にはKeyAsciiの値を0にして、引数をクリアにして、不要な処理が行われないようにしておきます。

```
Private Sub Text1_KeyPress(KeyAscii As Integer)
    If KeyAscii = 13 Then
        InputBtn_Click
        KeyAscii = 0
    End If
End Sub
```

Lesson35 ・・・・A1(Q&A-35.vbp)

①為替レートを表示していたテキストボックスコントロールを削除し、リストボックスコントロールを配置します。

テキストコントロールを削除し、リストボックスコントロールを配置する

② ［List］プロパティに、125円から135円までの金額を入力します。

ここに入力する

③フォームのLoadイベントプロシージャで、リストボックスの先頭の項目を選択状態にします。

```
List1.ListIndex = 0
```

④ ［換算］ボタンのClickイベントプロシージャで、換算式の分母に、リストボックスコントロールでクリックされた金額を組み込みます。
リストボックスコントロールの項目が選択されると、［ListIndex］プロパティに、その項目番号が格納されます（リストの先頭は「0」です）。そこで、リストボックスの［List］プロパティの引数に、［ListIndex］プロパティを組み込むと、選択された項目名を取り出すことができます。これを、換算式に当てはめます。

```
Money = CLng(Text2.Text) / CLng(List1.List(List1.ListIndex)
```

これで、リストボックスから選んだ為替レートで、計算式が実行されます。

⑤何回もレートを変えて換算計算をできるように、［換算］ボタンのClickイベントプロシージャにある、

```
Text2.Text = ""
```

という式を削除しておきます。

⑥これで、完成です。換算したい円の金額を入れ、リストボックスからレートを選んで［換算］ボタンをクリックすると、そのレートで計算が実行されます。レートは何回でも変更して、換算結果を確認できます。

為替計算プログラム － Excelを起動中です
現在の為替レート ：1ドル 125
¥ 3000 換算
$ $24.00 終了

最初は￥125で換算する

為替計算プログラム － Excelを起動中です
現在の為替レート ：1ドル 131
¥ 3000 換算
$ $22.90 終了

次にリストから￥131を選んで［換算］ボタンをクリックと、入力した金額はそのままで換算結果が変更される

Lesson36 ････A1(Q&A-36.vbp)

リストボックスの項目がクリックされたときに、コードの撮影データやイメージファイルの表示処理と同じ場所に、7列目のセルの内容をステータスバーコントロールに表示する処理を追加します。

```
StatusBar1.Panels(1).Text = .Cells(Pos, 7)
```

実際には、これをステータスバーコントロールのパネルの［Text］プロパティに設定すると表示されます。

Excel - photoalbum.xls

E	F	G
日	撮影場所	コメント
97年8月	羽田空港	旅行前の空港のロビーを撮影
96年7月	清里	清里の駅前にあった時計のついた建物
96年8月	八ヶ岳高原	八ヶ岳の牧場で白馬のポニーと記念撮影
95年6月	自宅	自宅で咲かせたバラ。品名は「プレイボーイ」で四季咲きのフロリバンダ
97年8月	北海道	北海道に旅行に行った時に撮影

7列目のセルが
コメントのセル

Custom Photo Album

アルバムを開く　終了

写真名
馬と子供
バラ－プレイボーイ
窓と花

カメラ　Canon A1
レンズ　85mm + クローズアップフィルタ
撮影日時　95/06/01
撮影場所　自宅

自宅で咲かせたバラ。品名は「プレイボーイ」で四季咲きのフロリバンダ

リストボックスを
クリックすると…

ステータスバーにコメント
が表示される

索引

数字

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

W

あ

か

さ

た

な

は

ま

や

ら

わ

● 著者紹介 ●

瀬戸　遥（せと　はるか）

フリーソフトの作者からパソコン関係のフリーライターに転身。主に、パソコン通信・ホームページ作成とVisual Basic関連の解説書を中心に執筆、これまで「10日でおぼえる」シリーズの他、約40冊の書籍を翔泳社他多くの出版社から出版している。
現在、月刊「Visual Basic Magazine」（翔泳社）に、VBAのテクニック解説記事を連載中。

●カバー装丁　株式会社アレフ・ゼロ（宇田俊彦）

● 「10日でおぼえる」のホームページ ●

本書の内容確認には十分気を付けておりますが、万一CD-ROMに収録したファイルや本文の誤記などで訂正が発生した場合には、以下のホームページにて訂正内容をお知らせします。ご利用下さい。

http://www.shoeisha.com/book/hp/10days/

10日（とおか）でおぼえる
Visual Basic（ビジュアルベーシック） 6.0 実践教室（じっせんきょうしつ）

1999年11月25日　初版第1刷発行
2002年4月10日　初版第6刷発行

著　者　VBテックラボ＆瀬戸 遥
発行人　速水浩二
発行所　株式会社 翔泳社
印刷・製本　株式会社 廣済堂

＊落丁・乱丁はお取り替えいたします。弊社営業部（03-5362-3810 / kanrika@shoeisha.co.jp）までご連絡ください。

＊本書へのお問い合わせについては、iiページに記載の内容をお読みください。

ISBN4-88135-825-1　Printed in Japan

10日でおぼえる

●ご利用の前に、
本書viiiページの
「CD-ROMをご使用の前に」、
およびCD-ROMに収録の
ファイル「README.TXT」を
必ずお読みください。

●本CD-ROMに
Visual Basic 6.0は
収録されておりません。

SE
SHOEISHA

COMPACT
disc

CD-ROM for
Windows95/98/NT4

Visual Basic 6.0 実践教室

VBテックラボ
&
瀬戸 遥 著

カバー写真
『ルナ』
 Photographed by Nob Fukuda

ISBN4-88135-825-1

C3055 ¥3200E

9784881358252

株式会社 翔泳社

定価：本体 3,200円＋税

1923055032002

第1日 かんたん表計算ソフトの作成
- 1時限目 ユーザーインターフェイスの作成
- 2時限目 合計計算処理の作成
- 3時限目 セルデータをファイルに保存する
- 4時限目 ファイルからデータを読み込む
- 5時限目 自習

第2日 ワープロソフトの作成
- 1時限目 ツールバーコントロールの組み込み
- 2時限目 リッチテキストボックスコントロールの配置
- 3時限目 文字属性の操作
- 4時限目 自習
- 5時限目 自習

第3日 便利なコントロールを使う
- 1時限目 スライダーコントロールを使う
- 2時限目 プログレスバーコントロールを使う
- 3時限目 ステータスバーコントロールを使う
- 4時限目 アップダウンコントロールを使う
- 5時限目 自習

第4日 時計の作成
- 1時限目 アニメーションクロックの作成
- 2時限目 デジタルクロックの作成
- 3時限目 タスクバーでの時刻表示（アイコンの点滅）
- 4時限目 自習
- 5時限目 自習

第5日 マルチドキュメントビューワの作成
- 1時限目 MDIプログラムの作成
- 2時限目 ユーザーインターフェイスの作成
- 3時限目 コードの作成
- 4時限目 自習
- 5時限目 自習

第6日 インターネットプログラムの作成
- 1時限目 ホームページダウンロードプログラムの作成
- 2時限目 FTPプロトコルを使う
- 3時限目 リストビューコントロールを使う
- 4時限目 簡単FTPクライアントプログラムの作成
- 5時限目 自習

第7日 カスタムWebブラウザの作成
- 1時限目 インターネットエクスプローラを操作する
- 2時限目 Internet Explorer Control Padの作成
- 3時限目 カスタムWebブラウザの作成
- 4時限目 自習
- 5時限目 自習

第8日 Win32APIを使う
- 1時限目 APIの機能と特徴
- 2時限目 APIビューアを使う
- 3時限目 メモリチェッカーの作成
- 4時限目 自習
- 5時限目 自習

第9日 APIを使ったアプリケーションの作成
- 1時限目 バージョン情報表示システム
- 2時限目 壁紙チェンジャーの作成
- 3時限目 自動壁紙チェンジャーの作成
- 4時限目 ディスクの空き容量を把握する
- 5時限目 自習

第10日 ExcelをVisual Basicで操作する
- 1時限目 Excel操作の基本知識
- 2時限目 ファイルからオブジェクトを作成する
- 3時限目 ワークシート関数を使う
- 4時限目 為替換算プログラムの作成
- 5時限目 フォトアルバムの作成

10日でおぼえる Visual Basic 6.0 実践教室

VBテックラボ & 瀬戸 遥 著

SE SHOEISHA